

Learning *Online* Network with CAPA

Author's Tutorial And Manual

June 26, 2014

LON-CAPA Group

Michigan State University

Contents

1	Introduction to Authoring in LON-CAPA	6
1.1	About This Manual	7
1.2	Login as Content Author	7
1.3	Roles Screen	8
1.4	Menu Options	8
1.4.1	Inline Menu	8
1.4.2	Remote Control	9
1.5	Resource Types	10
1.6	Description of Authoring Space	13
1.7	Searching Existing Resources	15
1.8	Browsing Existing Resources	17
2	Help	17
2.1	Online Help	17
2.2	Where to Find Additional Help	18
2.3	Requesting New Features and Submitting Suggestions	18
3	HTML Page Overview	19
4	Problem Types in LON-CAPA	19
4.1	Problem Types	19
4.1.1	Foils	19
4.2	Radio Response Problems	19
4.3	Option Response Problems	19
4.3.1	Option Response Problems with Concept Groups	19
4.3.2	Example: Concept Group	20
4.3.3	Example: Matching Problem	20
4.4	String Response Problems	21
4.5	Numerical Response Problems	21
4.6	Formula Response Problems	21
4.7	Math Response Problems	22
4.8	Custom Response Problems	22
4.9	Function Plot Response Problems	22
5	Authoring Content in LON-CAPA	22
5.1	Authoring and Editing Content Pages	22
5.1.1	How to Edit Existing Content Pages	24
5.2	Authoring Problems Using LON-CAPA	24
5.3	General Problem Editing	25
5.3.1	Authoring Surveys	27
5.3.2	Adding Picture	27
5.3.3	Dynamic Plots	27
5.3.4	Importing Testbanks	28

5.3.5	Answer Display Overview	28
6	Authoring Radio, Option, String Response Problems	30
6.1	Authoring Radio Response Problems	30
6.1.1	Randomization	31
6.2	Authoring Option Problems	31
6.3	Simple Option Response: No Concept Groups	33
6.4	Authoring a String Response Problem	33
7	Authoring Numerical, Formula, Custom Response Questions	35
7.1	Authoring Numerical Response and Formula Response Problems	35
7.2	The Parts of a Numerical Response Problem	35
7.3	Simple Numerical Response Answer	39
7.4	Simple Script Usage	39
7.4.1	Variables in Scripts	40
7.4.2	Variables in the Text Block	40
7.4.3	Variables in the Answer Block	41
7.5	Calling Functions	41
7.5.1	Numerical Response Randomization	42
7.6	Authoring Formula Response Problems	42
7.7	Dynamic, Randomized Problems: Putting It All Together	43
7.8	Units, Format	43
7.9	For More Information	44
7.10	Formula Response	44
7.10.1	Sample Specifications	44
7.10.2	Formula Notes	45
7.10.3	Example Formula Response	46
7.11	Authoring Math Response Problems	46
7.12	Custom Response Problems	48
8	Authoring Dynamically Generated Plots	50
8.1	Specifying Curves to Plot	52
8.1.1	Plotting Data Points	53
8.1.2	Plotting Functions	53
8.1.3	Data and Line Styles	54
8.2	Plot Labels and Key	55
8.3	Plot Axes Details	56
8.4	Color Selection	58
9	Authoring Function Plot Response Problems	59
9.1	Introduction	59
9.2	Elements	59
9.3	Rules	61

10 Using Libraries	66
10.1 Authoring Library Files	66
11 Authoring Adaptive/Conditional Hints	68
12 Publishing Your Resources	77
12.1 What is Metadata?	77
12.2 Publishing a Resource	77
13 Printing Your Resources	80
13.1 Printing from Authoring Space	80
13.2 Printing a Subdirectory of Problems	80
13.3 Tips for Improving Print Output	81
13.3.1 TeXsize Attribute	81
13.3.2 TeXwidth Attribute	82
13.3.3 TeXDropEmptyColumns Attribute	83
13.3.4 Image TeX Attributes	83
13.3.5 TeX Type Attribute	84
13.3.6 TeX Itemgroup Width Attribute	84
13.3.7 TeX Layout Attribute	85
13.4 Troubleshooting PDF Errors	85
14 Authoring Maps: Sequences and Pages	86
14.1 Authoring Sequences	86
14.2 Authoring a Simple .sequence With The Simple Editor	87
14.3 Authoring a Simple .sequence With The Advanced Editor	88
14.4 Page Maps	91
14.5 Courses: Top-level Sequence	91
15 Tags Used in XML Authoring	92
15.1 Response Tags	93
15.1.1 numericalresponse	93
15.1.2 imageresponse	93
15.1.3 optionresponse	94
15.1.4 radiobuttonresponse	94
15.1.5 dataresponse	94
15.1.6 externalresponse	94
15.1.7 Attributes For All Response Tags	95
15.2 responseparam and parameter	95
15.3 Foil Structure Tags	96
15.4 Hint Tags	96
15.5 Input Tags	97
15.6 Output Tags	97
15.7 Internal Tags	102
15.8 Scripting Tags	102

15.9 Structure Tags	103
16 <script> Tag	104
16.1 Supported Script Functions	104
16.2 Script Variables	106
16.3 Table: LON-CAPA Functions	107
16.4 Table: CAPA vs. LON-CAPA Function Differences	115
17 Bridge Task	119
17.1 Introduction to Bridge Task	119
17.2 Bridge Task Features	120
17.3 Authoring Bridge Task	121
17.4 Bridge Task XML Editing	122
17.4.1 .Task Headers	122
17.4.2 .Task Parameter and Variable	123
17.4.3 .Task Questions and Criteria	124
17.4.4 .Task Finishing Up	127
17.5 Bridge Task Edit Mode	127
17.5.1 Introductions	128
17.5.2 Questions and Criteria	129
17.5.3 Parameter and Variable	131
17.5.4 Edit Mode Finishing Up	133
17.6 Setting Up a Bridge Task	133
17.6.1 Bridge Task and Slots	134
17.6.2 Bridge Task and Conditional Resources	134
17.7 Handing In Bridge Task Files	135
18 Appendix: Symbols in TeX	136
18.1 Greek Symbols	136
18.2 Other Symbols	138
19 Appendix: Physical Units	139

1 Introduction to Authoring in LON-CAPA

LON-CAPA is a web-based content management system that helps to organize and present your course website, deliver and manage assignments, and manage student enrollment, assessment, and grading. Typically all author functions will be completed using a web browser (Firefox, Chrome, Safari, IE or similar). The one exception to this is where your LON-CAPA domain has been configured to support webDAV access, in which case you may be able to carry out standard file operations (copy, move, add file etc.) using your computer's standard filesystem interface, after you have established access to your authoring space volume.

LON-CAPA has three work spaces: the ROLES menu, the course/community space where courses are developed from resources, and the authoring space where resources are composed and published. There are two user manuals for LON-CAPA, a course coordinator manual and author manual. Also, there are quick reference guides to building a course and building an exam, available at <http://help.loncapa.org>. This is the author manual used to create resources such as problems that can later be added to your course.

Before creating problems, you should have:

- developed learning objectives that you want to evaluate for your course/community
- determined the appropriate question formats and developed your problems for input into LON-CAPA. Examples for question formats are provided in this manual and also when authoring a problem.
- developed the directory structure that you plan to use to organize your resources

Overview of the Authoring Process

Graphics, problems, and html pages are all considered **resources**. Additional resources include reusable snippets of perl, xml, cascade style sheets, etc. This manual documents the process used to create and organize the more advanced types of resources.

The authoring process involves these steps:

- create or upload a resource. The resource can combine other uploaded resources such as graphics, code snippets, or problem sequences
- test and revise your resource
- publish the resource to make it available for integration into a course/community and/or sharing
- revise your resources after publishing to improve clarity or eliminate bugs

Importance of Planning your Directory Structure

Once a resource has been published, the published version can never be moved or deleted. Thus, it is important to plan your folder structure. Old resources can be marked obsolete, and the version in your authoring space deleted, but the published version(s) will remain in your folders in the locations in which they were originally published.

Understanding xml and the Colorful Editor

Problems are written in xml markup, which can appear complex when you first start to work with LON-CAPA if you have not done html or other coding. However, each xml element has a starting and closing argument, just like html. This manual includes a reference on xml markup used to write problems.

The authoring environment includes a 'colorful' web-based editor that can be used for authoring your first problems. Even experts will often start with the colorful editor to build a template for a problem. It is possible to switch back and forth between the colorful editor and xml when composing and testing your problems. Often simple issues are easier to clean up directly in xml. Also, it is easier to copy/paste xml to build additional problem sections compared to using the colorful editor.

Scripts

The power of LON-CAPA for problem randomization and computing randomized answers is realized through writing perl script at the top of a problem. Example scripts are included in many example problems, and most resource authors publish scripts with the problems, so many examples are available. Many special functions have been created for use in scripts. This manual includes a section on writing scripts.

Maxima and R

Two computer algebra systems are interfaced to LON-CAPA, Maxima and R. This provides for algebra and calculus problems and responses. The R system has strong capabilities for statistics. Special script functions are provided to call Maxima and R to generate correct responses for a randomized problem, and also to check student responses.

Comments

Commenting your xml and scripts is important for both you and other users. Commenting within a loncapa/perl script is denoted with a `#`. This can be entered anywhere in a line and the remainder of the line will be ignored when parsing. Problems are coded in xml. Comments in xml are of the format `<!-- comment -->`. However, it is important to know that the XML comments propagate through to the rendered web page viewed by students while the perl comments within the script are not. Hence, writing solution hints within XML comments is discouraged for obvious reasons.

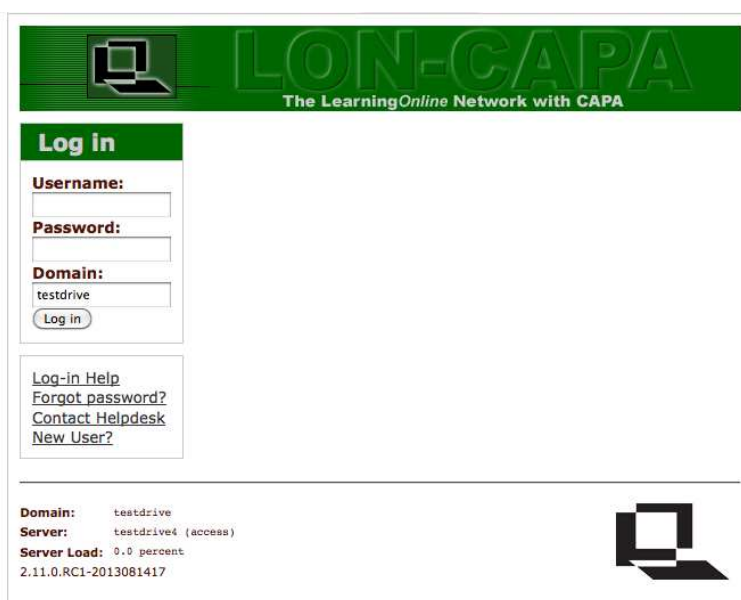
1.1 About This Manual

Throughout this manual, keywords and phrases literally present on the computer screen will be referred to in **bold type**. Function names and scripts will be shown in a **typewriter font**.

1.2 Login as Content Author

To begin using LON-CAPA, you first need to log in to your account on LON-CAPA. Open your web browser and navigate to your local LON-CAPA URL. You will be presented with a log in screen.

Fill in the Username and Password boxes with your information. Then press the Login button. This will take you to your LON-CAPA User Roles menu.



LON-CAPA
The LearningOnline Network with CAPA

Log in

Username:

Password:

Domain:
testdrive

[Log-in Help](#)
[Forgot password?](#)
[Contact Helpdesk](#)
[New User?](#)

Domain: testdrive
Server: testdrive4 (access)
Server Load: 0.0 percent
2.11.0.RC1-2013081417

Figure 1: LON-CAPA Log in screen

Note: Your Username and Password will be given to you by your system administrator. Both are case sensitive, so make sure you type them with the correct case.

1.3 Roles Screen

After logging into LON-CAPA with your username and password, you will see a screen where you can select your **ROLE**. LON-CAPA is a roles-based system. You can always use the same login even though your roles may change. You can be an instructor in one class, a TA for another class, and an author of resources – all with the same username. Selecting **Roles** at the top right of a page, or **Switch to another user role** on the Main Menu page will bring you back to the **Roles** screen.

Two main types of Roles exist in LON-CAPA: Authoring Roles, and Course Roles. Authoring roles are used to develop learning resources. The course role is where resources are inserted into courses. You may have multiple instances of either type of role. For example, each separate course has a course role. You may make use of more than one authoring space role depending on your involvement in collaborative LON-CAPA content creation.

1.4 Menu Options

1.4.1 Inline Menu

Author Inline Menu

The default menu system is the **Inline Menu**. Links are available near the top of screens.



Figure 2: Author Remote Control

1.4.2 Remote Control

Remote Control

LON-CAPA also offers a *Remote Control* menu. The Remote Control is a separate pop-up browser window, and is automatically sized and placed in the upper left of the screen. To enable and experiment with the Remote Control menu, navigate to: Main Menu > My Space > Set my user preferences > Content Display Settings > Launch Remote Control. To exit from the remote, either close the window, or look for the inline menu command to exit on the upper right of the parent browser window.

Hints for each button appear when hovering over the buttons. Click the words *Extended Display* immediately above the remote command display to obtain a list of hints.

The commands available on the remote change depending on whether you are in a authoring role or course role, and, in the case of a course role, change depending on the functionality available to your current role for the screen currently displayed in the main browser window. Some of the important commands on the Remote Control are:

- **ROLES (CHOOSE ROLE)** opens the roles screen for you to select which user role to assume.
- **CCAT (COURSE CATALOG)** opens search for courses/communities.
- **CUSR (CHANGE USER PRIV)** add/manage user privileges.
- **AUTH (AUTHORING SPACE)** displays the authoring space for your account.
- **RES (RESOURCE SPACE)** allows you to browse the LON-CAPA network directory. (Searching resource space is usually preferred unless you know where to look for a particular resource.)
- **COM (COMMUNICATION)** allows you to access the communication functions in the system.

- **SRCH (SEARCH LIBRARY)** brings up a screen that lets you search the LON-CAPA resources using multiple criteria.
- **PREF (PREFERENCES)** brings up a screen that allows you to change some preferences.
- **EXIT (LOGOUT)** will log you out of the LON-CAPA system.

1.5 Resource Types

LON-CAPA provides several types of resources for organizing your course website. Resource behavior is determined by file extension. Valid extensions are: .xml, .html, .xhtml, .htm, .xhtm, .problem, .page, .sequence, .rights, .sty, .task, .library, .js, .css, .txt.

- A **.css** cascading style sheet file can be published for use in multiple courses. Use of a specific CSS style within a resource can be set using an html tag within a text block that refers to a class by name, where the class is included in the published css file. You would specify use of the css file with a <link> tag, and indicate a file dependency with an allow tag (if the <allow> tag is not specified LON-CAPA should add it at the first publication step).

```
<startouttext />
<link rel="stylesheet" index="11"
href="/res/yourdomain/yourusername/yourpath/testcss.css" type="text/css" />
<span class="Yellow">After the Earth and Mars, which other body in our
solar system is thought to be the next most likely site for finding
life?</span><endouttext />
<allow src="/res/yourdomain/yourusername/yourpath/testcss.css" />
```

where testcss.css contains:

```
.Yellow {
  color: yellow;
}
```

It is also possible to indicate use of a css file within a course, course-wide, for a specific folder, or specific resource etc, by setting the parameter: CSS file to link [Part: 0] (cssfile).

- A **.html** HTML file, (formerly known as a Content Page), displays course content. It is a conventional HTML page. These resources use the extension “.html”. By using the “New File..” dropdown, you can enter a file name with one of the other extensions, htm, xhtml, xhtm, xml. For more information see the section 3.

Although the extensions: xhtml and xhtm imply that the file should be xhtml compliant (i.e., be a valid document, according to w3c validation meaning (amongst other things):

all elements closed; attributes case-sensitive; attributes required by certain elements, certain specials characters need to be included as entities in some attributes etc.), LON-CAPA does not test whether a document with an xhtml extension is actually valid xhtml.

Files of type .xml are supported for legacy files, but it is suggested that new files are one of the html varieties. See also the .library file description for xml.

- A **.problem** Problem resource represents problems for the students to solve, with answers stored in the system. These resources are stored in files that must use the extension “.problem”. Problems are coded in a combination of Perl and xml markup tags. Most of this manual concerns authoring problems. For an overview see the section 4.1
- A **.page** Page is a type of **Map** which is used to join other resources together into one HTML page. For example, a page of problems will appears as a problem set. These resources are stored in files that must use the extension “.page”.

For performance purposes it is best not to include too may resources in a .page (8 would be a good limit; and likely fewer if maxima or R are required for computation). The Course Editor offers a similar tool called a “Composite page” which resides within a course rather than in the published repository.

The underlying XML structure, and behavior of a .page file are the same as used for a “Composite page”, which can be added to a course using the Course Editor. One difference between the two is that for a .page in Authoring Space, one of the Authoring Space editors (Simple Edit or Advanced Edit) will be used to modify the contents of the file, whereas for a Composite Page, the Course Editor is used.

Options for reuse of a course Composite page are: (a) use cloning to copy everything from the old course to the new; (b) as course coordinator, copy the Composite map into the Course Editor’s clipboard, change role to course coordintator in a different course, and past the item into the other course from the Course Editor’s clipboard.

The course manual describes ways to combine resources in folders, which is an alternative to creating composite pages. A potential advantage of a a composite page is that the browser’s inbuilt capabilities can be used to print all resources included within a page with a single “Print” call. That said, LON-CAPA’s own print utility which creates PDFs allows students to print all resources in a folder to a single PDF, which lessens the advantage that accrues from use of a composite page. The disadvantage of use of a published .page is that once a .page is published, reordering or modifying the contents requires use of the resource assembly tool in authoring space, followed by republication. Publishing a .page in the repository is suggested if the content does not need to change.

For more information on .pages, see the section 14.4

- A **.js** javascript file contains javascript code which might be referenced with HTML in a textblock in a LON-CAPA problem in a manner similar to a .css file.

- A **.library** library file contains LON-CAPA XML which can be imported into other .problem files. Typically a .library file will hold commonly used subroutines, or data structures (to be called in LON-CAPA perl script blocks). It is included in a problem in the colorful editor by using the “Import a File” selection in a dropdown list in the colorful editor.
- A **.sequence** sequence is a type of **Map** which is used to link other resources together. The users of this resource can use links to follow the sequence. Sequences are stored in files that must use the extension “.sequence”. Sequences can contain other sequences and pages.

A published sequence contains XML which describes which LON-CAPA resources are to be grouped together in a folder. If a .sequence file is imported in its entirety then there is limited control over which specific resources are shown, and in which order. The “Randomorder” and “Randompick” checkboxes in the Course Editor can be used to cause the resources to be displayed in a randomorder (randomized for each user), and to display M of the total N resources in the sequence (again the choice of which M are shown is randomized for each user). By contrast, if the “Import from Assembled Map”, a “Select Map” link will allow all resources in the published sequence to be imported into the current folder, as distinct resources, allowing them to be reordered, and cut/removed, as preferred using the standard Course Editor tools.

Conditional sequences are published sequence files for which conditions have been specified for one or more links between resources in the map. These have to be created using the “Advanced Edit” button when creating a new sequence file in Authoring Space. These are the types of sequence for which a student’s progress through a series of resources can be specified. Grading of conditional sequences can be complicated.

- A **.rights** file is used to specify custom access rights for a published resource. The access rights to apply to a specific resource are specified on publication (or republication) of the resource. A .rights file also must be published to be selectable during resource publication as a “Custom Distribution File” when “Customized right of use ...” is selected from the “Copyright/Distribution” dropdown list. See the “Publishing A Resource” section 12.2.
- A **.sty** is a LON-CAPA style file used to apply custom styling to specific tags in a LON-CAPA problem. The styles defined in the file can be applied to display of all resources in a course using:

Main Menu->Modify course configuration->Display of resources (checked) + click “Display” button->Click “Select Style File” link (Default XML style file item) to launch window to select published .sty file, then click “Save Changes” in main window.

Style files can contain different instructions for different targets (e.g., web or tex). For example, the following style file would replace the <h1>-tag in all incorporated pages by the annoying <blink>-tag:

```
<definetag name = "h1">
```

```

    <meta></meta>
    <render>
        <web><blink></web>
        <tex>\section{</tex>
        <latexsource>\section{</latexsource>
    </render>
</definetag>

```

```

<definetag name = "/h1">
    <meta></meta>
    <render>
        <web></blink></web>
        <tex>}</tex>
        <latexsource>}</latexsource>
    </render>
</definetag>

```

When printing, LON-CAPA will use the tex `\section` as `<h1>`-rendering.

Note that `.css` is perhaps a more modern way to achieve formatting.

- A **.task** task file is a bridge task file. Bridge tasks permit assessment using rubrics. See the Bridge Task section 17.

1.6 Description of Authoring Space

The Authoring Space is the environment in LON-CAPA where you create and update your course resources. Creating content involves both *authoring* the resource during editing, and then *publishing* the resource to make it available for use in courses. The figure 3 shows the top menu. If you move your mouse pointer over your name at the top left of the page, a dropdown box will be displayed which you can use to select “Preferences” (see Setting Preferences in the course coordination manual).

The displayed menu also includes:

- **Main Menu** provide links to additional personal settings.
- **Authoring Space** returns you to the current authoring space.
- **Browse** provides a link to published resources.
- **People** provides a menu to add coauthors.

The gray bar under the menu provides a clickable bread crumb trail. The current **authoring space** folder location is displayed near the bottom of the figure 3, and is clickable. The **Recent** dropdown box lists recent folder locations for quick navigation.

The upper right menu is not shown here, but includes the following links:



Figure 3: Authoring Space Upper Menu

Type	Actions	Name	Title	Status	Last Modified
	Go to ...	Parent Directory			Tue Jun 25 02:57:53 pm 2013 (EDT)
	Select action ▾	example folder			Tue Jun 25 04:12:42 pm 2013 (EDT)
	Select action ▾	example.problem (Edit) (EditXML) (Clean Up)	Edit Metadata	Unpublished	Tue Jun 25 03:44:05 pm 2013 (EDT)
	Select action ▾	my_file.html (Edit) (Clean Up)	Edit Metadata	Unpublished	Tue Jun 25 03:36:17 pm 2013 (EDT)

Figure 4: Example of Author Directory

- **Message** links to your e-mail box.
- **Roles** returns you to your roles page.
- **Help** provides a link to online help for the authoring space. The online help is a modified version of this manual.
- **Logout** is a link to close the session.

At the bottom of the page is a directory as shown in the figure 4. On the left edge, icons help identify the content type. The **Actions** dropdown provides commands to move, copy, paste, publish the resource or other content-type-dependent options. The **Name** column provides a link to the resource or folder. The **Title** column provides a link to the meta data provided to help other users find your resource. The **Status** column shows whether the resource is **Published**, **Unpublished**, or **Obsolete**. The vertical bar (pink in the figure) color denotes the status: Green (published); Pink (unpublished); Gray (obsolete).

In the center of the page are three action menus: **Actions for current directory**, **Upload a new document**, **Create a new directory or LON-CAPA document**. The first two menus are intuitive when exploring them. The third menu is shown in the figure 5 and new content types are described later in this manual.

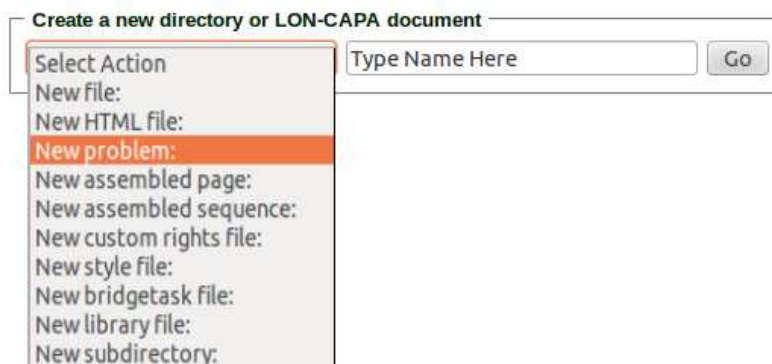


Figure 5: New Content Menu

1.7 Searching Existing Resources

The Basic search page allows you to specify a phrase or expression to search for in the LON-CAPA resource repository. Search strings can contain quoted phrases, multiple words, '-' (meaning “not”), and multiple words or phrases joined with 'or'. For example:

- **cat** Matches resources about cats.
- **cat or dog** Matches resources about cats or dogs.
- **cat dog** Matches resources about both “cat” and “dog”.
- **“fluffy cat” -dog** Matches resources on the phrase “fluffy cat” that do not contain “dog”.

There are two checkboxes on the basic search page. The first allows you to limit your search to the domain of the machine you are currently logged into. The second causes a thesaurus lookup of your search phrase. Search results will include resources which match any of the related phrases as well as the original phrase.

The resource metadata fields searched in a basic search are listed below. It is worth noting that the data searched are self-reported by the resource author. In many cases, authors are unable to provide relevant values for the fields searched. If you need to search based on properties other than the following, use the 1.7 Advanced Search link on the basic search page.

- title
- author
- subject
- notes

- abstract
- keywords

The Advanced Search in LON-CAPA allows you to search specific metadata fields for matching resources. Resources which match all of the constraints specified will be returned.

For many of the search fields, a “related words” checkbox is available. This causes each search term or quoted phrase in the search for that field to be matched in our thesaurus. Resources which matches in the given field any of the search phrases (depending on the logic in the original search phrase) will be returned.

The following fields can be searched:

- Title
- Author
- Publisher/Owner
- Author Space
- Last Modifying User
- Keywords
- Notes
- Abstract
- Standards
- Mime type
- Lowest grade level
- Highest grade level
- Domain
- Copyright
- Language
- Creation and Modification Dates

Dynamic metadata, computed based on resource usage statistics, can be searched as well. The following fields can be searched, with a minimum or maximum value being specified:


- Network-wide number of accesses (hits)
- Total number of students who have worked on this problem

- Average number of tries till solved
- Degree of difficulty
- Degree of discrimination

Evaluation data may be searched, with a minimum or maximum value being specified:

- Material presented in clear way
- Material covered with sufficient depth
- Material is helpful
- Material appears to be correct
- Resource is technically correct


1.8 Browsing Existing Resources

Click  **Browse published resources** on the Main Menu to browse the LON-CAPA resource pool. This is the same interface used when importing a document into your course documents except that you only have permissions to view the resources, not select them for importing. Note that at the top of the Resource Space Screen there are some options. For example, selecting "Title" will display the titles from the metadata from each problem. Selecting "All versions" will show all versions of the resource which are saved whenever the resource was republished.

The resources are organized by the domain, then the username. The **Recent** dropdown box provides a shortcut to recent browsing history.

2 Help

2.1 Online Help

On many LON-CAPA pages, you will see one or more small blue circles (each with a white question mark inside) . These blue circles are links to relevant help documentation. Each LON-CAPA help page also includes a search box, in which you can enter keyword(s) to search all LON-CAPA help documents.

Also, a **Help** link is available in the upper right. This help page provides links to the course manual (from within a course) or the author manual (from the authoring space) and provides both an online version and pdf version. You may wish to save a pdf version of each so that you can reference the help for one role while working in the other role. The help page also provides a link to request help from the LON-CAPA help desk and a link to report bugs (Course Coordinators and Authors only). In some cases you may also see a green square containing a white question mark alongside the text: "FAQ". This provides a link to relevant help item in the user-contributed documentation found in the LON-CAPA FAQ-o-matic system.

2.2 Where to Find Additional Help

You can find additional help about LON-CAPA and download copies of this manual at <http://help.lon-capa.org>. For additional help, use the **Help** link on the upper right of the screen to find a link to contact the helpdesk, or search the loncapa listservs at <http://mail.lon-capa.org/mailman/swish.cgi>. You may wish to join the listserv loncapa-users at <http://mail.lon-capa.org>. The family of users is very helpful in assisting users with questions posted to the listserv.

2.3 Requesting New Features and Submitting Suggestions

We Welcome Your Ideas and Contributions...

LON-CAPA is open-source software where everyone is free to contribute to the development. If you think of an enhancement, or find a bug in the system, please report it. LON-CAPA uses Bugzilla (<http://www.bugzilla.org/>) to report and track bugs and enhancement requests. All reported bugs and enhancement requests posted on Bugzilla are examined by the core LON-CAPA team. Having the request in Bugzilla helps keep track of both bugs and feature requests, and is the preferred method of reporting bugs and requesting enhancements. You'll be able to see how the developers respond to the bug, which developer is currently responsible for it, and what the developer's intentions are.

You must have an account to use Bugzilla. You can create an account at <http://bugs.lon-capa.org> by clicking on the link **Open a New Bugzilla Account**. You will be directed to send an e-mail to the LON-CAPA Helpdesk to request creation of a Bugzilla account. Once you have an open account, please search existing cases for related content before submitting a new case. Select "LON-CAPA" in the "Product" category when searching. Hold the Ctrl key while clicking multiple items. The "Component", "Version", and "Target" can be left undesignated. If you find a related bug, you can add comments instead of reporting a new bug. If you find a bug that you are interested in following, you can add your e-mail to the "Add cc:" box and click "Commit".

You can report a bug or request an enhancement as follows:

1. Open the web page <http://bugs.lon-capa.org>
2. Click on the **Enter a new bug report** link.
3. Select the (LON-CAPA) product link.
4. At the login screen, enter your e-mail address and password.
5. Click the [Login] button.
6. Select the LON-CAPA Version, Component, Platform and Operating System (as much as known).
7. Enter a summary in the "Summary" text box.
8. Enter a description in the "Description" text box.

9. Click the [Commit] button.

A developer will follow up with you.

3 HTML Page Overview

HTML files are HTML documents that display the course information you are presenting. They have been previously called Content Pages.

You may create HTML files and then upload them, or you may create HTML files within the Authoring Space.

4 Problem Types in LON-CAPA

4.1 Problem Types

In this manual we will cover basic types of problems: Radio Response, Option Response, String Response, Numerical Response, Formula Response, and Math Response. You will need to identify which types of problem you want to use and create appropriate questions for your course.

4.1.1 Foils

In the LON-CAPA system, a **Foil** is the statement next to the drop-down box or radio button in a Radio Response or Option Response problem. Foils do not need to be text; they can be images or other resources.

4.2 Radio Response Problems

Radio Response problems present a list of foils with buttons. The student can select *one* of these statements by clicking the appropriate radio button.

4.3 Option Response Problems

Option Response problems present foils to the student with drop-down boxes. The student can select the matching choice for the foils from a list of choices. Optionally, the foils may be bundled into Concept Groups and the system will select one foil from each group to display to the student.

By default, the list of options is presented in front of the foils. Using the optional `<drawoptionlist />` tag, the list of options can be embedded into the foil.

4.3.1 Option Response Problems with Concept Groups

Each **Option Response** problem can have three parts:

1. The Concept Groups

Edit
EditXML
Random Seed: 1023307703
Change
Reset Submissions
☐ Show All Foils

Enter question text here.

This is statement ThreeA of concept Three. True
 This is statement OneC of concept One. True
 This is statement FourB of concept Four. False
 This is statement TwoA of concept Two. True

Submit Answer

Tries 0/2

Figure 6: Option Response Problem

- The options for the students to select, by default “True” and “False”
- The hint for the student

Each **Concept Group** has some number of foils representing questions which are conceptually related. Option Response Problem Templates are available for 4 and 8 Concept Groups. When the Option Response problem is presented to a student, the LON-CAPA system will randomly select one foil from each Concept Group and present it to the student. In order to receive credit for the problem, the student must select the corresponding option from the drop-down box for each given foil.

4.3.2 Example: Concept Group

A Concept Group may contain the following True/False questions:

- “Mark Twain” is the pen name of Samuel Clemens.
- Mark Twain wrote “The Call of the Wild”.
- Mark Twain wrote “Huckleberry Finn”.
- Mark Twain spent most of his life in the Congo.

For each foil, the author marks it **true** or **false**. When the student logs on and attempts to answer this question, the student will see only one of the four choices for that Concept Group. They then go on to do the remaining three to seven Concept Groups in this question before submitting their answer.

4.3.3 Example: Matching Problem

Option Response problems can be used as matching problems.

For example, you might want to ask the student to match musical compositions with their composers. You could create an Option Response problem with 4 Concept Groups, and place the following four foil groups each in its own concept group:

- Claire de Lune, Ballade (Debussy)
- The Pastoral Symphony, The Ninth Symphony (Beethoven)
- Sleeping Beauty Suite, The Dance of the Sugar Plum Fairies (Tchaikovsky)
- Slavonic Dances, New World Symphony (Dvorak)

You could then add the following options to the option list:

- Debussy
- Beethoven
- Schubert
- Tchaikovsky
- Dvorak

The same answers can be used more than once, or not at all, as you see fit. It is conventional to place such a warning in the **Text Block** describing the problem to the students.

4.4 String Response Problems

String Response problems allow the student to submit a string of characters for the answer. Examples of String Response questions are vocabulary tests, short answers and chemical formulas.

Note that it is easy to abuse String Response problems. For instance, consider the question “Who wrote ‘Huckleberry Finn’?” If you tell the system the answer is “Mark Twain”, and a student answers “Twain”, the system will mark it wrong. If they answer “Samuel Clemens”, then the student will definitely get it wrong. There is some room for flexibility in the string processing, but it can be difficult to get it all correct. Before you use a String Response problem, be sure you can easily characterize correct answers.

4.5 Numerical Response Problems

Numerical Response problems are answered by entering a number and (optionally) a unit, such as 2.5 m/s^2 . Tolerance and required significant digits can be specified as well.

4.6 Formula Response Problems

Formula Response problems ask the student to type in a formula as an answer. If the answer is $2x^2 + 4$, the student is allowed to type “ $2*x*x+4$ ”, “ $x*x + x*x + 4$ ”, “ $2*x^2 + 4 - 10$ ”, or any other equivalent expression. Formula Response problems have many of the same characteristics of Numerical Response problems, including the ability to run scripts, dynamically generate answers, etc.

4.7 Math Response Problems

Math Response is a way to have a problem graded based on an algorithm that is executed inside of a computer algebra system based on an algorithm written by the problem author. It is extremely powerful, as it tests answers for conditions rather than agreement with a particular correct answer. An unfortunate byproduct, however, is that it cannot be analyzed by several of the LON-CAPA statistics tools.

4.8 Custom Response Problems

Custom Response is a way to have a problem graded based on an algorithm constructed in the perl script. The use of this response type is generally discouraged, since the responses will not be analyzable by the LON-CAPA statistics tools.

4.9 Function Plot Response Problems

A **Function Plot Response** problem requires that the student create a plot that matches specified criteria. Examples can be functions that have certain slopes, curvature, maxima or minima at specified independent coordinate values. The students create their answer by dragging the curves and adjusting the slopes.

5 Authoring Content in LON-CAPA

LON-CAPA internally recognizes the types of resources by the filename extension. Within the authoring space, there are two methods to create a new resource:

- If you are using Inline Menus, you may use the dropdown menu on the right of the authoring screen to indicate the type of resource you want to create as shown in the figure 5. The filename extension will be specified when you select the resource type. This method is best for new authors.
- Or, you may type the full url of the new resource in URL bar of your browser. You must use a valid file extension. LON-CAPA will recognize the new resource type by the filename extension and present you the appropriate menus to create your resource. This method is appropriate for advanced authoring.

5.1 Authoring and Editing Content Pages

Many users use stand-alone editing programs such as Dreamweaver to create HTML files. To upload HTML files generated with such tools, you can use the **Browse...** button in the Authoring Space, locate your HTML file, and use the **Upload File** button to upload a HTML file in LON-CAPA. Remember to upload any local graphics your generated web pages may have referenced using similar procedures. The links to graphics in your html page must match the relative locations for the uploaded files.

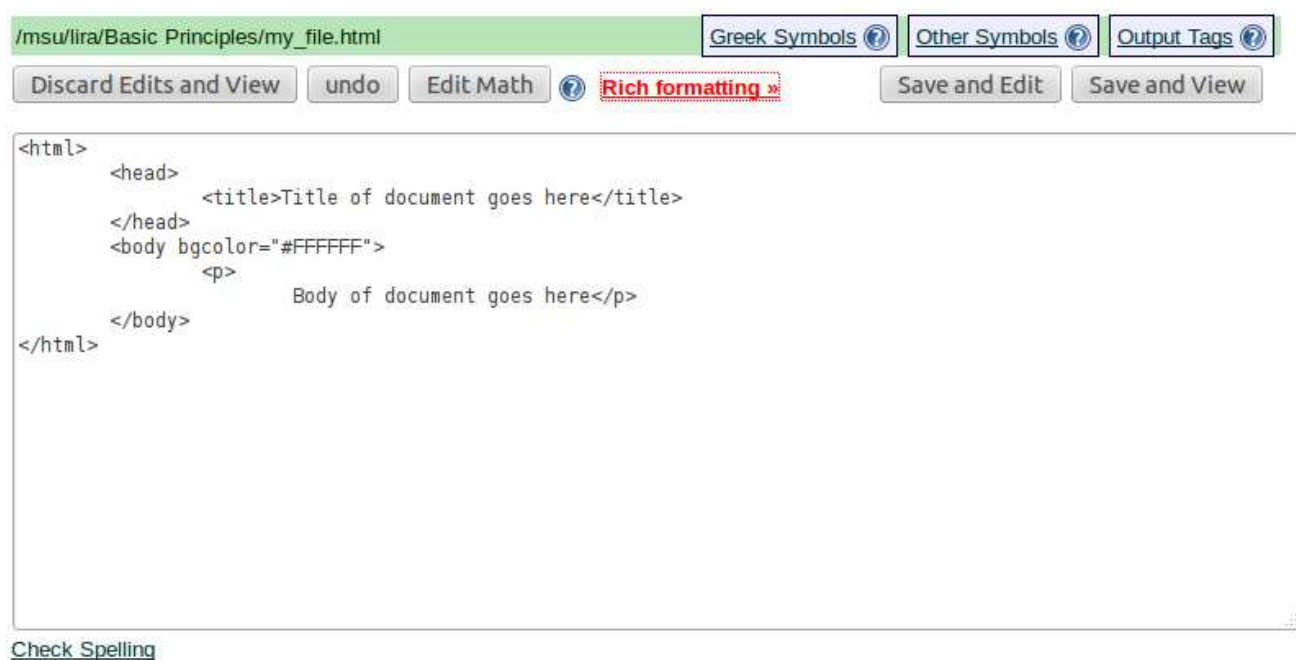


Figure 7: Plain Text HTML Editor

To create a new html page using the Inline Menus, browse to the desired folder using the folder navigation links in the lower part of the authoring screen. Then select 'HTML file' from the new content dropdown menu on the right of the authoring space page as shown in the dropdown menu of the figure 5.

When you edit a web page, you will be presented with a plain text editor as shown in the figure 7. Clicking the link for the Rich Formatting displays the HTML WYSIWYG editor with buttons to help you create html entities with correct tags as shown in the figure 8. If the rich layout is not as you expect, look for the button to view/edit the source, and then manually make adjustments to the html tags. Note that buttons are available to select graphics to include in the HTML file. Buttons are also provided to create/edit links.

To create equations, you can use the 'Edit Math' button in the plain text editor, or you can type LaTeX equations directly in the $\langle m \rangle \langle /m \rangle$ tags in the plain text editor. The 'Edit Math' button launches a pop-up window containing the DragMath equation editor (Java plugin support in user's browser required). The pop-up includes a "Render LaTeX" button which can be clicked to insert the math expression composed in Drag Math as LaTeX within $\langle m \rangle \$ \$ \langle /m \rangle$ in the file editor in the original window.

Note that convenient help links for **Greek Symbols**, **Other Symbols**, and **Output Tags** immediately above the editor in the figures. Also note the link to check spelling below the editor.

The URL of the page in the root of your space would look something like *http://(your library server)/priv/domain/username/new_resource.html* .

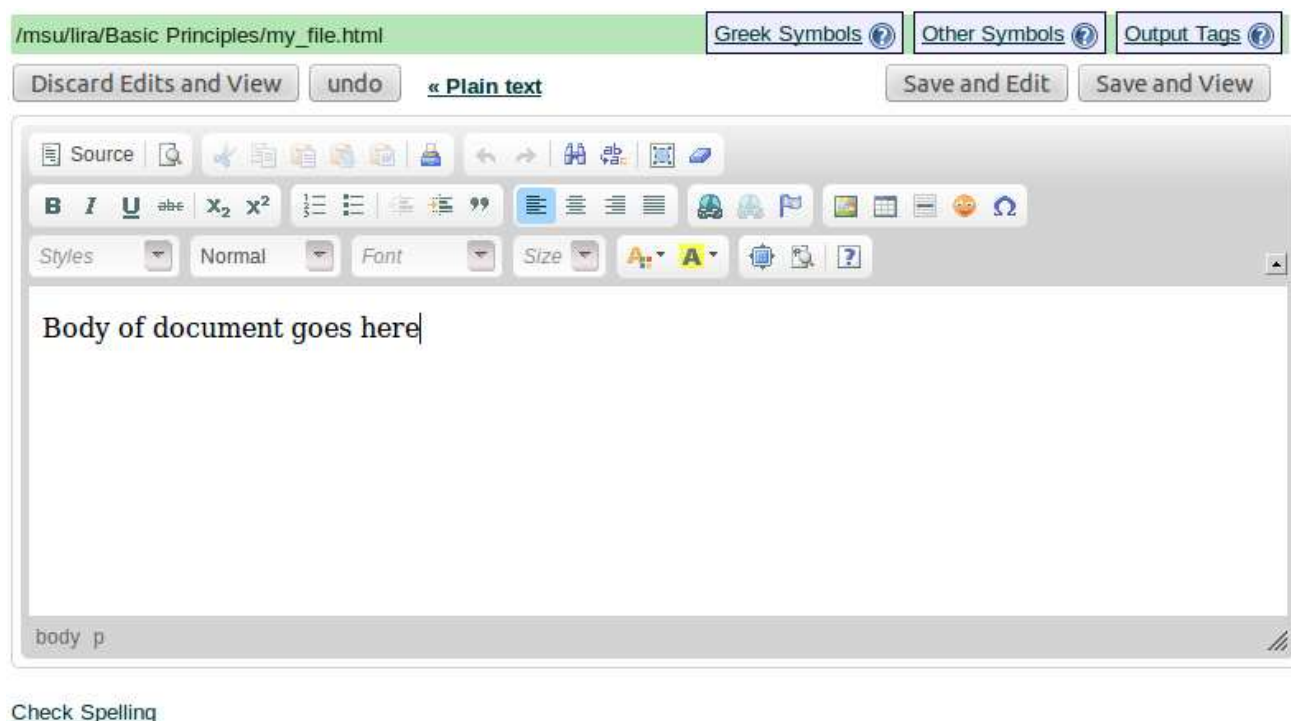


Figure 8: Rich Text HTML Editor

5.1.1 How to Edit Existing Content Pages

You may edit any Content Pages that have been created.

To edit Content Pages:

1. Navigate to the folder with the HTML file.
2. Click on the link for the name of the HTML file to edit.

5.2 Authoring Problems Using LON-CAPA

When you create a new problem as in the figure 5, LON-CAPA will offer to you example templates that are highly recommended for beginners. A subset of the problem templates is illustrated by the multiple choice selections shown in the figure 9. Click the **Example** link next to menu item to see an example problem. Once you make a selection, you can use the template and edit the content to convert the template into your own problem. By default, the template problem will be displayed in the problem testing mode with a header shown in the figure 10. The problem statement (not shown here) appears below the testing menu. The menu will be described later. Click the **Edit** button in the lower left to open the colorful editor. The EditXML button is for advanced users.

The menu for the colorful editor is shown in the figure 11. The button functions are:

- **Discard Edits and View** - return to the testing menu without saving



Figure 9: Multiple Choice Problem Menu

- **EditXML** - switch to the XML Editor
- **Undo** - forget last edits
- **Save and Edit** - Save and continue editing in the colorful editor
- **Save and View** - Save and return to the testing menu

An example of the colorful editor is shown in Figure 13. Each box in the colorful editor has a purpose, and the boxes are colored by their purpose. Note that the current editor includes more space between the blocks than shown in many of the figures in this manual. Also, the newer editor has an 'Insert' dropdown box between blocks so that you can insert content easily.

While several problem types are covered in this manual, in LON-CAPA all problems involve similar XML elements. The menu for the XML editor is shown in the figure 12. Note that the editor includes help links at the top. The **Edit** button starts the colorful editor. The **Edit Math** button adds latex math. The other buttons are the same as the colorful editor buttons. If you find yourself uncertain about the name or function of an LON-CAPA xml tag, consult the reference in section 15.

The problem editor gives you a testing area where you can try your problems out, with several different randomizations by varying the **Random Seed**. If you answer a problem correctly and can no longer enter new answers, you can get the answer field back by hitting the **Reset Submissions** button.

Problems also may appear differently after the answer date, or on an exam. The editor provides dropdown selectors to view problems with such conditions set.

5.3 General Problem Editing

The following capabilities are available in all problem types:

Problem Testing ?

Problem Status: Problem Type: ☐ Show All Foils

Feedback Mode: Apply style file: [Select](#)

Language: Math Rendering:

[Change View](#) [Show Default View](#) [Reset Submissions](#)

[New Randomization](#) [Change Random Seed To:](#) [Calculate answers](#) for versions. ?

[Edit](#) [EditXML](#)

Figure 10: Problem Testing Menu

Problem Editing ?

[Discard Edits and View](#) [EditXML](#) [Undo](#) [Save and Edit](#) [Save and View](#)

Figure 11: Colorful Problem Editor Menu

Problem Editing ? [Script Functions](#) ? [Greek Symbols](#) ? [Other Symbols](#) ? [Output Tags](#) ?

[Discard Edits and View](#) [Edit](#) [Undo](#) [Edit Math](#) ? [Save and EditXML](#) [Save and View](#)

Figure 12: XML Problem Editor Menu

5.3.1 Authoring Surveys

Surveys are created by authoring a conventional problem, such as a Radio button or Option problem. During the authoring process, you will need to select one of the foils as “correct”. Which foil is set to “correct” will not matter, however; that setting will be ignored when you implement the survey.

Within the course environment, to add a survey to the course, insert the “problem”, and then set the parameter for “**Question Type**” to “Survey”, “Survey with Credit”, or “Anonymous Survey”. To evaluate the survey, you will perform analysis using Reports -> Survey Results or Statistics and Analysis -> Detailed Problem Analysis.

The results examined with either tool will be displayed sorted by foil name, so take some care while authoring to name the foils. For example “Foil10” appears before “Foil2” alphabetically, so your survey may be easier to interpret if you insert a zero before single digits, such as “Foil02” and “Foil10” which will sort as expected.

5.3.2 Adding Picture

To add a picture to a problem, the picture must first be uploaded to your authoring space, then published. Then, in the text area of your problem, add the following:

```

```

where DOMAIN is the domain the picture is in, AUTHOR is the person who published the picture, and the rest is the standard path to the picture.

It is also possible for advanced users to use a script variable in the place of the picture URL, like this:

```
<img src='$picture' />
```

and use the string variable \$picture in the script of the problem to select from several possible pictures. If you do this, you will need to **Edit XML** for the problem and add the various graphics used in the problem to the `<allow_` tags on the bottom.

When print resources with pictures, LON-CAPA will automatically convert graphics in EPS files. (EPS is a graphics format designed for printing.)

The automatic conversion of a web graphic to an EPS file will sometimes look blocky, because paper has a much higher resolution than the web. If you would like to provide LON-CAPA with an EPS file to use while printing for a given graphic file, upload your EPS file into your authoring space with the same name as the .gif, .jpg, or other web graphic, except ending with the extension “.eps”. When you publish the file, LON-CAPA will automatically use it in place of the web image file when printing.

For instance, if you have a graphics file `my.image.gif`, you can upload an EPS file named `my.image.eps`.

5.3.3 Dynamic Plots

Dynamic Plots can be generated from calculations based on random numbers. The student can be asked to respond to the plot. Another option is to use student input to display a graphical result to students based on an input that they provide.

5.3.4 Importing Testbanks

The following requirements must be met to ensure that you will succeed in building LON-CAPA problem files from a file containing testbank questions.

1. The uploaded testbank file containing questions and answers must be either plain text, rich text format (RTF), or a web page (i.e., HTML format). Testbank files in RTF and HTML formats may contain images. If conversion to LON-CAPA fails for RTF or HTML formats, re-saving the testbank file from the application originally used to generate it in plain text, and re-uploading the file to LON-CAPA may solve the problem (albeit with the loss of formatting information).
2. All questions must occur before any of the answers. Each question should begin on a new line starting with a number followed immediately by a space, a period, or enclosed in parentheses, i.e., 1 , 1., (1), 1), or (1 . If you use a word processor to create the testbank questions you must disable auto-formatting of lists, otherwise the formatting will not conform to the new line requirement.
3. One or more correct answers need to be provided for all questions (although the answer text may be blank for *essay* questions). Each answer should begin on a new line using the same numbering scheme as used for the questions, and all answers must occur after all the questions.
4. *Multiple choice* and *multiple answer correct* questions should consist of (i) the question number followed by (ii) a question stem beginning on the same line and (iii) two or more foils, with each foil beginning on a new line and prefixed by a unique letter, or Roman numeral, listed in alphabetic or numeric order, beginning at a (alphabetic) or i (Roman numeral), followed by a period, or enclosed in parentheses, i.e., a., (a), i., or (i)
5. If *fill-in-the-blank* or *multiple answer* questions have more than one correct answer, each answer should appear in a comma-, tab-, space-, or new line-delimited list. For a *ranking/ordering* question, the "answer" should contain the foil identifiers correctly ordered in a similarly delimited list. If two or more foils have the same ranking, they should occur together, with an equals sign separating equally ranked foils [e.g., (b),(e)=(a),(d),(c)]

5.3.5 Answer Display Overview

Several xml tags are available to provide student guidance during problem display and also after the answer date. These include `<notsolved>`, `<preduedate>`, which are useful when the students are working a problem, and `<solved>`, `<postanswerdate>`, which are helpful in providing feedback. The content in the xml tag `<postanswerdate>` is suppressed until after the answer date.

Answers are typically not displayed to the student when they submit an answer to a problem, unless their answer is correct. When the problem is assigned in a course, a parameter for the 'answer date' must be set such that students will see answers displayed when

they revisit the problem after that date. The answer date is distinct from the due date which can also be set when a problem is assigned.

Suppose you want the answer displayed as soon as the student finishes the problem rather than waiting until the answer date. For example, this may occur in a multipoint problem where you want the students to see the correct answer for one part before proceeding to the next part. If you would like the computer answer to be displayed after the student answer is correct, or the maximum number of tries has been reached, this can be achieved by setting the parameter `problemstatus = answer`. Using the colorful editor, the parameter can be inserted WITHIN the problem part, but BEFORE the response tag (e.g. `numericalresponse`). For example:

```
<part id="01">
<parameter name="problemstatus" id="11" type="string_problemstatus"
  default="answer" description="Show Problem Status" />
<numericalresponse answer="$c" id="01a">
<!-- using no tolerance since exact integer is expected for answer -->
  <textline readonly="no" spellcheck="none" />
</numericalresponse>
</part>
```

The options for the `problemstatus` are: “yes” - shows correct or incorrect only; “answer” - shows computer answer as described above; “no” - don’t show correct/incorrect feedback; “no_feedback_ever” - suppresses all feedback.

If you want to provide additional explanation in addition to the answer as soon as the maximum tries allowed for the part have been reached, include a conditional block in the problem part, and set the condition for the part to evaluate to true when the problem is either correct, no tries remain, or it is post-answer date.

To achieve this behavior, add a call to the LON-CAPA function `&check_status('partid')` for each part to the script block, and set a scalar to the value returned for each part, then use that scalar in the conditional block, e.g.,

```
<script type="loncapa/perl">
# for part: a
$status_a = &check_status('a');
# other computations
</script>

<part id="a">
<block condition="$status_a">
<startouttext />
<br />
Some explanation text about the answer, $answer
<endouttext />
</block>
</part>
```

Figure 13: Radio Response Creation Form

6 Authoring Radio, Option, String Response Problems

6.1 Authoring Radio Response Problems

To create a **Radio Response** problem, create a new resource as described in section 5. This is a “problem” resource so the URL must end in “.problem”. You should see a screen as in figure 5. You will need to specify the question text and foil statements.

1. Select a radio response template from the menu. You will be presented a form such as in the figure 13.
2. In the **Text Block** at the top of the problem, remove the sample text and type the question for your problem. Ex: “What is two plus two?”
3. Locate the **Response: One of N statements** element. In the **Max Number of Shown Foils** text box, place the number of foils you wish to display to the student.
4. Locate **Foil 1**. Remove the text that is in the text box and put the *correct answer* for the problem in the **Text Block**. For example, “Four.” Make sure this is set to **true** in the **Correct Option** field.

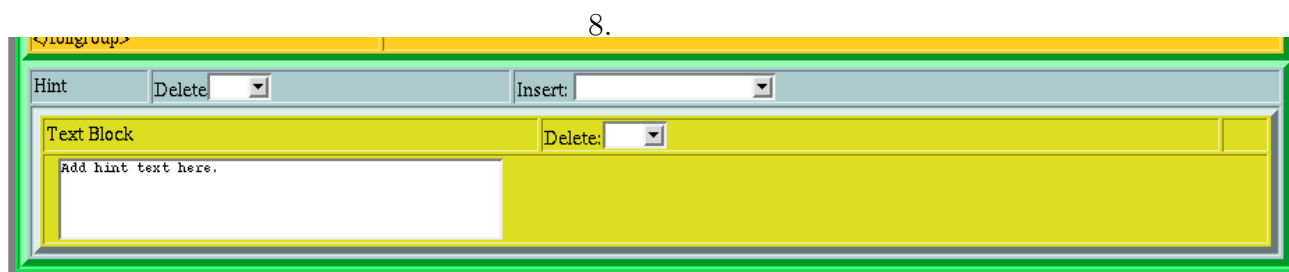


Figure 14: Hint Element

5. Below it, you will see **Foil 2**. Remove the text in the text box and put an *incorrect answer* for the problem. Ex: “Purple.” Make sure this is set to **false** in the **Correct Option** field.
6. Repeat the previous step until you’ve filled in all of the other incorrect answers you wish to offer the students.
7. Once you’ve filled in all of the incorrect answers, delete any extra foils or change the **Correct Options** on the other foils to **Unused**.
9. Scroll down to the Hint element. See the figure 14 Type some text that will help students when they answer incorrectly. You may delete the hint by selecting **Yes** from the **Delete** drop-down box.
10. Click the **Submit Changes** button located at the top of the frame. If you do not do this, your changes will not be saved.

The **Correct Option** drop down box controls whether or not a given answer will be accepted as a correct answer. If it is set to **true**, that answer will be considered a correct answer. Any number of foils can be marked **true**, but only one will be shown to any given student. If it is set to **false**, it will be considered an incorrect answer. If it is set to **Unused**, the system will not use that foil.

6.1.1 Randomization

LON-CAPA will randomize the choices presented to each student and the order they are presented. If you wish to present a random set of incorrect answers, create more foils than you wish to display, and then set the **Maximum Number of Shown Foils** to a value less than the number of foils created. If you wish to present each student the same choices, make sure the **Maximum Number of Shown Foils** box contains a number greater or equal to the number of foils that you created, which will force them to all be displayed.

6.2 Authoring Option Problems

To create an Option Response problem, create a new resource as described in section 5. This is a “problem” resource so the URL must end in “.problem”. You should see a screen as in figure “Option Response Editor”.

Figure 15: Option Response Editor

1. In the drop-down option box as seen in figure 5, select **Option Response Problem with N Concept Groups**, where N is the number of Concept Groups you wish the problem to have, and click the **New Problem** button.
2. Click the **Edit** button above the sample problem to enter edit mode. You should see the Option Response page open up.
3. Replace the text in the **Text Block** with text that explains the conditions for your problem.
4. Locate the **Max Number of Shown Foils** element and type a number from 1 to 8 to display that number of questions. You cannot display more than one foil from each concept group, so this option will only reduce the number of foils displayed, if it is less than the number of concept groups in your Option Response problem.
5. Now you must define the options the students can select. For each option you wish to add to the Option Response question, type the option into the **Add new Option** box in the **Select Options** section, then hit the **Save Changes** button. If you do not hit the **Save Changes** button, your option will not be selectable below. (You can delete unwanted options in the last step.)
6. Now, you need to define the question foils. Look for the foil with the name “One”. Type the question into the text box and select the correct option for that question

from the **Correct Option** drop-down menu. Click **Submit Changes** to save this question foil. Repeat this step for all remaining foils.

7. Locate the foils that are not being used. In their **Delete** menus, set the value to **Yes**. Once you've set the Delete menu value correctly for all the foils, click the **Save Changes** button.
8. In the Hint area, provide a helpful hint for users who get the problem incorrect, and click the **Save Changes** button.
9. Make sure all the options you want to delete are not used for any of your foils. If a deleted option is used in a foil, it will appear in a text box in the **Correct Option** area for that foil. To make the drop-down box reappear, type an option already defined in the **Select Options** field, and hit **Submit Changes**. A drop-down box will reappear. To delete the irrelevant options from the Option Response question, select that option from the **Delete an Option** drop down, and hit the **Save Changes** button. Do this for each option you wish to remove.

6.3 Simple Option Response: No Concept Groups

If you select **Simple Option Response** from the drop-down box, you will get a template that will allow you to enter up to eight foils with no grouping. The system will randomly mix these foils when presenting them to the student. You can have more foils than the **Max Num of Shown Foils** so that each student will not have the identical foils.

6.4 Authoring a String Response Problem

To create a **String Response** problem, create a new resource (described in 5.2. This is a “problem” resource so the URL must end in “.problem”).

1. In the drop-down option box as seen in the figure 5, select **String Response Problem**, and click the **New Problem** button.
2. Click the **Edit** button above the sample problem to enter edit mode. You should see the String Response editor page open up, which should look something like what you see in the figure 16.
3. Clear the text from the **Text Block** at the top of the problem, and type in your problem.
4. In the **Answer Box**, type the correct answer.
5. Select the answer condition from the drop-down. There are three cases to choose from:
 - (a) **cs**: This means “Case Sensitive”. For example, this is useful in chemistry, where HO and Ho are completely different answers. The student must match the case of the answer.

View

EditXML

undo

Submit Changes

Insert:

Text Block

Delete:

The 3 types of string answers are:

cs: Case Sensitive

ci: Case Insensitive

mc: Multiple Choice, Order of characters unchecked.

The answer is NaCl and it is case sensitive.

Response: String

Delete

Insert:

Answer: NaCl

Type: cs

Single Line Text Entry Area

Delete

Size:

Hint

Delete

Insert:

Text Block

Delete:

Add hint text here.

Submit Changes

Figure 16: String Response Editor

- (b) **ci**: This means “Case Insensitive”. The system does not use the case of the letters to determine the correctness of the answer. If the correct answer is “car”, the system will accept “car”, “CAR”, “Car”, “caR”, etc.
- (c) **mc**: This means “Multiple Choice”. The student’s answers must contain the same letters as the question author’s, but order is unimportant. This is usually used to give a multiple choice question in the question’s **Text Block**, which may have several correct parts. If the author sets the correct answer as “bcg”, the system will accept “bcg”, “cbg”, “gcb”, etc., but not “bc” or “abcg”.

It is conventional to inform the students if the problem is case sensitive, or that the order of the answers doesn’t matter.

6. Optionally, locate the **Single Line Text Entry Area** block and set a length in the Size box. This will only affect the size of the box on the screen; if you set the box size to 2, the student can still enter 3 or more letters in their answer.
7. Scroll down to the **Hint** element, and type some text that will help students when they answer incorrectly, or delete the hint by setting the **Delete** field to **Yes**.
8. Click the **Submit Changes** button.

7 Authoring Numerical, Formula, Custom Response Questions

7.1 Authoring Numerical Response and Formula Response Problems

Numerical Response problems are answered by entering a number and an optional unit. For instance, a numerical response problem might have an answer of $2m/s^2$. Formula Response problems are answered by entering a mathematical formula. For instance, a formula response problem might have an answer of $x^2 + 11$. The answer may be in any equivalent format. For instance, for $x^2 + 11$, the system will also accept $x * x + 11$ or $x^2 + 21 - 10$.

Numerical Response problems are very powerful. In fact, they are so powerful it would be impossible to fully explain what is possible in a simple document. This chapter will focus on getting you started with Numerical Response problems and show you some of the possibilities, with no prerequisite knowledge necessary. The more you learn, the more you will find you can do.

If you like, you can follow this chapter as its own tutorial. Create a Numerical Response problem using the instructions in the “Authoring Content in LON-CAPA” section 5, ending your resource name with “.problem”, and create a new **Simple Numerical Response** problem.

7.2 The Parts of a Numerical Response Problem

A Numerical Response problem has seven major parts by default:

Script		Delete: <input type="button" value="v"/>	
<div>#Enter the computations here</div>			
Text Block		Delete: <input type="button" value="v"/>	
<div>What is 2 + 2?</div>			
Response: Numerical		Delete: <input type="button" value="v"/> Insert: <input type="button" value="v"/>	
Answer: <input type="text" value="4"/>		Unit: <input type="text"/> Format: <input type="text"/>	
Parameters for a response		Delete: <input type="button" value="v"/>	
Name: <input type="text" value="tol"/>	Type: <input type="text" value="tolerance"/>	Description: <input type="text" value="Numerical Tolerance"/>	Default: <input type="text" value="5%"/>
Parameters for a response		Delete: <input type="button" value="v"/>	
Name: <input type="text" value="sig"/>	Type: <input type="text" value="int_range,0-16"/>	Description: <input type="text" value="Significant Figures"/>	Default: <input type="text" value="0,15"/>
Single Line Text Entry Area		Delete: <input type="button" value="v"/>	
Size: <input type="text"/>			
Hint		Delete: <input type="button" value="v"/> Insert: <input type="button" value="v"/>	
Text Block		Delete: <input type="button" value="v"/>	
<div>This should be easy!</div>			

Figure 17: Numerical Response editor

1. The **Script** is the heart of advanced Numerical Response problems. It can be used to decide some of the parameters of the problem, compute the answer to the problem, and do just about anything else you can imagine. The Script language is **Perl**. You do not need to know Perl to use the **Script** block because we will be stepping through some advanced examples in this chapter, but knowing Perl can help.
2. Like other problem types, the **Text Block** is used to display the problem the student will see. In addition, you can place variables in the **Text Block** based on computations done in the **Script**.
3. The **Answer** is the answer the system is looking for. The answer can use variables calculated/defined in the problem's **Script** block, allowing the answer to be determined dynamically (including randomization).
4. A **tolerance** parameter determines how closely the system will require the student's answer to be in order to count it correct. The tolerance will default to zero if it is not defined. The tolerance parameter should always be defined for a numerical problem unless you are certain only integer answers are generated from your script and you want students to reply with exactly that integer.

If the computer answer is a floating point number, the tolerance should not be zero. Computers can only approximate computations involving real numbers. For instance, a computer's [decimal] answer to the simple problem $\frac{1}{3}$ is "0.3333333333333331". It *should* be an infinite series of 3's, and there certainly shouldn't be a "1" in the answer, but no computer can represent an infinitely long, infinitely detailed real number. Therefore, for any problem where the answer is not an integer, you *need* to allow a tolerance factor, or the students will find it nearly impossible to exactly match the computer's idea of the answer. You may find the default tolerance too large for some problems, so adjust as appropriate.

There are three kinds of tolerance. For some answer A and a tolerance T ,

- (a) an **Absolute** tolerance will take anything in the range $A \pm T$. So if $A = 10$ and $T = 2$, then anything between 8 and 12 is acceptable. Any number in the tolerance field *without* a % symbol is an absolute tolerance.
- (b) a **Relative** tolerance will take anything in the range $A \pm aT$, where T is interpreted as a percentage/100. Any number in the tolerance field *followed by* a % symbol is a relative tolerance. For example, $a = 10$ and $t = 10\%$ will accept anything between 9 and 11.
- (c) a tolerance that is a calculated variable (identified by \$ sign as the first character). For example, if an answer is $\$X$, and for a student possible values range from $-\$X1$ to $+\$X1$, you could choose $T = \$tolerance = \$2X1/100$; acceptable answers would then be from $\$X - \$tolerance$ to $\$X + \$tolerance$. (This is especially useful when answers close to zero are possible for some students)

Some care is necessary when setting the display format of the computer answer. Before testing the tolerance, LON-CAPA converts the computer answer, as generated in the script block, according to the format attribute in the numericalresponse tag.

Next, the formatted computer answer is "graded" relative to the significant figures parameter, if it is set (see section 5). If that test was passed, then a numerical comparison of the Computer's answer is made with the range of values:

$$(\$computerAnswer - \$tolerance) < \$formattedcomputerAnswer < (\$computerAnswer + \$tolerance)$$

If the `$formattedcomputerAnswer` satisfies the permitted range, then "correct" is returned for the computer answer. It is good idea to test multiple randomizations to make sure that your tolerance is compatible with the display format.

5. A **significant figures** specification is an optional setting that tells LON-CAPA how many significant figures there are required for the answer, as either a single number, e.g. 3, or a range of acceptable values, expressed as **min,max**. The significant figures field can be omitted if you do not want to constrain the number of significant digits in the student answers.

The system will check to ensure that the student's answer contains the required significant digits, useful in many scientific calculations. For example, if the computer answer is "1.3", and the problem requests three significant digits, specified by (entered without quotes) "3", the system will require the students to type "1.30", even though numerically, "1.3" and "1.30" are the same. A significant figure specification of (entered without quotes) "3,4" means both "1.30" and "1.300" are acceptable.

Authors should be clear in the problem statements to tell students the required number or range of significant figures. If the student response does not contain the correct number of significant digits, the LON-CAPA response will tell students to increase or decrease the digits in their response, but it will not tell them how many digits to use. These responses do not use up the number of trials, but such responses are frustrating for students. If you would like to ensure that at least three significant digits are used, then a specification such as 3,15 ensures at least three digits are used, but will quietly accept up to 15.

Note that care must be used when using formatted computer answers together with a significant digit specification. You must ensure that the formatted answer provides enough significant digits. To test the formatted answer, LON-CAPA converts the computer answer, as generated in the script block, according to the format attribute in the `numericalresponse` tag, e.g. 3f. Then LON-CAPA separately applies that number of significant figures to the computer answer, and if that result falls outside the range specified in the significant digit parameter, it "grades" the computer answer as `SIG_FAIL` (i.e., not correct). It is a good idea to use the problem testing environment to test plenty of different randomizations to make sure that your format and sig digits parameters are compatible. This helps ensure that the formatted answer has enough significant digits.

6. The **Single Line Text Entry** area, as in other problem types, allow you to manipulate the text entry area the student will see.

7. Finally, the **Hint** should contain text which will help the students when they answer incorrectly.

7.3 Simple Numerical Response Answer

Along with showing the Numerical Response editor, figure 17 also shows the parameters for one of the simplest possible types of numerical responses. The **Text Block** has the problem's question, which is the static text "What is $2 + 2$?" The **Answer** is "4". The **Hint** has been set to something appropriate for this problem. Everything else has the default values from when the problem was created.

If you create a problem like this, hit **Submit Changes**, then hit **View** after the changes have been submitted, you can try the problem out for yourself. Note the last box in the HTML page has the answer LON-CAPA is looking for conveniently displayed for you, along with the range the computer will accept and the number of significant digits the computer requires when viewed by an **Author**.

As you're playing with the problem, if you use up all your tries or get the answer correct but wish to continue playing with the problem, use the **Reset Submissions** button to clear your answer attempts.

7.4 Simple Script Usage

Totally static problems only scratch the surface of the Numerical Response capabilities. To really explore the power of LON-CAPA, we need to start creating dynamic problems. But before we can get to truly dynamic problems, we need to learn how to work with the **Script** window.

A script consists of several **statements**, separated by **semi-colons**. A **statement** is the smallest kind of instruction to the computer. Most problems will be built from several statements.

A script can contain **comments**, which are not interpreted as statements by the computer. Comments start with **#** and go to the end of that line. Thus, if a line starts with **#**, the whole line is ignored. Comments can also begin in the middle of a line. It is a good idea to comment more complicated scripts, as it can be very difficult to read a large script and figure out what it does. It is a *very* good idea to adopt some sort of commenting standard, especially if you are working in a group or you believe other people may use your problems in the future.

- One of the simplest statements in LON-CAPA is a **variable assignment**. A **variable** can hold any value in it. The variable name must start with a **\$**. In the **Script**, you need to assign to variables before you use them. Put this program into the **Script** field of the Numerical Response:

```
$variable = 3;
```

This creates a variable named **variable** and assigns it the value of "3". That's one statement.

Variable names are *case sensitive*, must start with a letter, and can only consist of letters, numbers, and underscores. Variable names can be as long as you want.

There are many variable naming conventions, covering both how to name and how to capitalize variables¹. It is a good idea to adopt a standard. If you are working with a group, you may wish to discuss it in your group and agree on a convention.

If you **Submit Changes** and **View** the problem, you will see nothing has changed. This is because in order for a variable to be useful, it must be used. The variable can be used in several places.

7.4.1 Variables in Scripts

Variables can be used later in the same script. For instance, we can add another line below the `$variable` line as such:

```
$variable2 = $variable+2;
```

Now there is a variable called `$variable2` with the the number “5” as its value.

Variables can also be used in *strings*, which are a sequence of letters. The underlying language of the script, Perl, has a very large number of ways of using variables in strings, but the easiest and most common way is to use normal double-quotes and just spell out the name of the variable you want to use in the string, like this:

```
$stringVar = "I have a variable with the value $variable.";
```

This will put the string “I have a variable with the value 3.” into the variable named “stringVar”.

If you are following this chapter as a tutorial, add the previous two lines to your **Script** and submit the changes for the problem. There’s no need to view it; there’s still no visible change.

7.4.2 Variables in the Text Block

Once you’ve defined variables in the **Script**, you can display them in the **Text Block**. For example, using the previous three-line script we’ve created so far, you can place the following in the **Text Block**:

```
See the 3: $variable<br />
See the string: <b>$stringVar</b><br />
```

If you save that and hit **View**, you should get what you see in figure 18. Note how the “`$variable`” was turned into a 3, and the “`$stringVar`” was turned into “I have a variable with the value 3.”

¹The author favors `capsOnNewWords`. Some people use `underscore.to.separate.words`. Many use up-percase letters to specify constants like `PI` or `GOLDEN_MEAN`. Some people always `StartWithCapitalization`. What’s really important is to be consistent, so you don’t have to guess whether the variable you’re thinking of is `coefFriction`, `CoefFriction`, `COEF_FRICTION`, or something else.

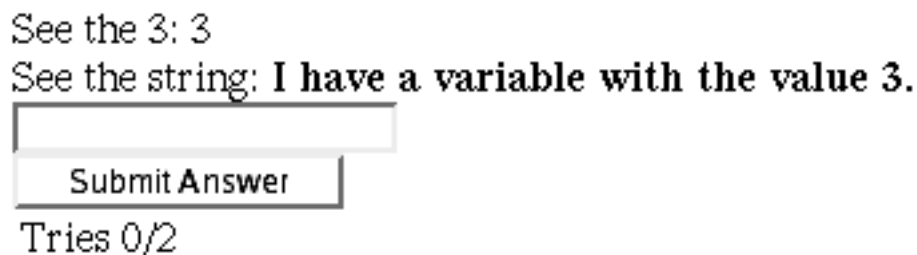


Figure 18: Result of Variables in the Text Block

```
$a = -3.0;
$b = &sin($a);
$c = &pow(3.0, &abs($a));
```

Figure 19: Some Function Calls

If you are generating numbers using a formula which includes a division operation, it is advisable to format your output. For example, if `$variable=1/3`, it will display with too many digits. You can adjust the format using the `num` output tag described in the Output Tags section 15.6

7.4.3 Variables in the Answer Block

You can use variables in the **Answer** part of the question. This means you can compute an answer to a question. If you set the answer of the question to be `$variable`, **Save Changes** and **View** it, you'll see that LON-CAPA is now expecting "3.0" as the answer, plus or minus 5%.

7.5 Calling Functions

With variables, you can store strings or numbers. **Functions** allow you to manipulate these strings or numbers. Functions work like mathematical functions: They take some number of arguments in, and return one argument, usually a number or a string for our purposes. There are a lot of functions available in LON-CAPA. You can see a complete list in the Script section 16.1 and LON-CAPA function section 16.3.

For now, let's just look at some simple examples.

In the **Script** block, function names start with `&`. Some example function calls are shown above. You can see that functions can take either variables, numbers, or the results of other function calls as parameters. The `&sin` function returns the sine of an angle expressed in radians. `&pow` raises the first parameter to the power of the second parameter. `&abs` returns the absolute value of the argument.

Script Delete? Script Functions

```
$k=&random(3,6,1);
$formula="a*x^$k";
$m=$k-1;
$derivative="$k*a*x^$m";
```

Insert:

Text Block Delete? Edit Math Greek Symbols Other Symbols Output Tags

Rich formatting »

What is the derivative of <tt>\$formula</tt> with respect to x?

Check Spelling

Insert:

Response: Formula Delete? Insert:

Answer: \$derivative Sample Points: ?

Pre-Processor Subroutine:

Single Line Text Entry Area Delete?

Size: 25 Click-On Texts (comma sep): Readonly: no

Insert:

Figure 20: Formula Response Problem

7.5.1 Numerical Response Randomization

If you're following this material and creating an example, click the button for a new random seed to generate other problem presentations.

7.6 Authoring Formula Response Problems

Formula Response problems authoring is illustrated by Figure 20. Note how the function is randomly generated, as well as the answer. Note that sampling is also available for evaluating answers. Sampling is a good alternative when students may give several equivalent forms that are difficult to compare in text. Clicking the help box next to the **Sample Points** field describes how to use sample points instead of a text comparison.

The screenshot shows the LON-CAPA problem editor interface with the following sections:

- Script** (light blue background): Contains the following code:


```
$slope1 = &random(1, 4, .1);
$slope2 = &random(-4, -1, .1);
$yint1 = &random(-10, 10, .1);
$yint2 = &random(-10, 10, .1);
$answer = ($yint2 - $yint1)/($slope1 - $slope2);
```
- Text Block** (yellow background): Contains the text:

For the lines defined by the following equations:
 $y = \text{slope1} x + \text{yint1}$
 $y = \text{slope2} x + \text{yint2}$
 At what value of x do these lines intersect?
- Response: Numerical** (green background): Includes a 'Delete' button and an 'Insert' dropdown menu.
- Answer** (green background): Includes fields for 'Answer' (containing '\$answer'), 'Unit' (empty), and 'Format' (empty).
- Parameters for a response** (pink background): Includes a 'Delete' button and a table of parameters:

Name	Type	Description	Default
tol	tolerance	Numerical Tolerance	.05

Figure 21: Slope Problem Parameters

7.7 Dynamic, Randomized Problems: Putting It All Together

Now you have all the tools to create those wonderful dynamic, randomized problems that you've seen in LON-CAPA. Consider a Numerical Response problem where the equations for two lines are randomly generated and the students are asked to find the intercept. Try filling out your problem with the parameters shown in Figure 21.

When creating randomized problems, you want to make sure that the problems always have an answer. Consider what might happen if two slopes are chosen, *both* with the expression `&random(-1.0, 1.0, .2)`. One out of ten students would get a problem where both slopes were equal, which has either no solution (for unequal y-intercepts) or an infinite number of solutions (for equal slopes and y-intercepts). Both of these cause a division-by-zero error on the division that computes the answer. There are many ways to avoid this, one of the easiest of which is picking one slope negative and one positive. This same problem can show up in many other places as well, so be careful.

7.8 Units, Format

Numerical Response problems can require units. In the problem editing form, place the desired unit in the **Unit** field. A complete list of units is in the Physical Units section (19). The computer will accept the answer in any of its accepted unit formats. For example, if the answer to a problem is “1ft”, the computer will accept “12in” as correct.

Additional units can be defined using the “customunits” parameter, which is a comma-separated list of unit conversions. For example

```
peck=2*gallon,bushel=8*gallon,gallon=4.4*L
```

would define the new units peck, bushel, and gallon. The last entry in a conversion chain always needs to be a unit that LON-CAPA already supports.

You can format the number displayed by the computer as the answer. For instance, if the answer is one-third, the computer will display that it computed “.33333333” as the answer. If you’d like to shorten that, you can use the **Format** field. Format strings like “2E” (without the quotes) will display three significant digits in scientific notation. Format strings like “2f” will display two digits after the decimal point. Format strings like “2s” will round a number to 2 significant digits. For a summary of numerical format statements, see the `num` tag item in the Output Tags section (15.6).

7.9 For More Information

The full power of Perl is well outside the scope of this document. Looking in the function list in the section 16.3 can give you some ideas. O’Reilly has some good Perl books. Enter “perl” in the search box at <http://search.oreilly.com>.

If you need help, you might search the listserv at <http://mail.lon-capa.org/mailman/swish.cgi> or consider joining the users listserv to ask other users for advice at <http://mail.loncapa.org>, or use the Help link to contact the Help desk. Often other users are quick to provide help.

Our advanced users often come to prefer the XML interface for the problems, available through the **EditXML** buttons. Covering the XML format is beyond the scope of this manual, but you can learn a lot by using the editor to make changes and seeing what happens to the XML. Supported XML tags are discussed in the Tags Used in XML Authoring section (15).

7.10 Formula Response

Formula Response problems have the same capabilities as Numerical Response problems, and add the ability to ask the student for a symbolic formula as an answer, instead of a simple number.

7.10.1 Sample Specifications

As you may know, it is extremely difficult to determine whether a given expression is exactly equal to another expression in general. For example, is $\sin 2x = 2 \sin x \cos x$? LON-CAPA has two ways of finding out if it is:

- algebraically, using a symbolic algebra system
- numerically, using sampling points

You need to determine which way is the safest in a given situation.

If you don’t specify sampling points, the symbolic algebra system is used.

If you do specify sampling points, LON-CAPA uses them. If your answer and the student’s answer agree at the sampling points within your given tolerance factor, the student’s answer will be accepted. If the student’s answer does not agree at the sampling points within your given tolerance factor, it will be rejected.

To specify where to sample the formulas for determining whether the student's answer is correct, you need to put a sampling specification in the **Sample Points** field. The sampling specifications take the following format:

1. A comma-separated list of the variables you wish to interpret,
2. followed by "@" (not in quotes),
3. followed by any number of the following two things, separated by semi-colons:
 - (a) a comma-separated list of as many numbers as there are variables, which specifies one sampling point, OR
 - (b) a comma-separated list of as many numbers as there are variables, followed by a colon, followed by another list of as many numbers as there are variables, followed by a #, followed by an integer.

The first form specifies one point to sample. The second form specifies a range for each variable, and the system will take as many random samples from that range as the number after the #.

For $2x^2 + 4$, with one variable "x", one could specify:

- "x@2", which will sample the answers only at 2. (This is generally a bad idea, as the student could get lucky and match at that point)
- "x@1:5#4" will takes 4 samples from somewhere between 1 and 5.
- "x@1:5#4;10" will takes 4 samples from somewhere between 1 and 5, and also sample at 10.

For $2x^2 + 3y^3 + z$, which has three variables, one could specify:

- "x,y,z@4,5,3;10,12,8#4;0,0,0", which take four samples from the box determined by the points (4, 5, 3) and (10, 12, 8), and also sample the point (0, 0, 0).

7.10.2 Formula Notes

- The formula evaluator can not handle things of the form " $x + - y$ ". If you have a random variable that may be positive or negative (as in the example following this section), you can try wrapping the references to that variable in parentheses. As always, it is a good idea to try out several randomized versions of your problems to make sure everything works correctly.
- **Never use relative tolerance in Formula Response problems.** Relative tolerance is poorly defined in Formula Response problems. Always use absolute tolerance.

7.10.3 Example Formula Response

A very simple formula response problem:

- In the **Script**, place the following:

```
$slope = &random(-5.0,5.0,.5);
$yint = &random(-5.0,5.0,.5);
$answer = "$slope*x + ($yint)";
```

- In the **Text Block**, place the following: “For a line with slope \$slope and y-intercept \$yint, what is y equal to?”
- In the **Answer**, place the following: \$answer
- Set the Tolerance to .000001.
- Set the **Sample Points** to x@0;1;2;3 .

7.11 Authoring Math Response Problems

Math response problems use a cas system to evaluate the student response. Which computer algebra system is to be used is specified in the cas argument of the mathresponse tag; both Maxima and R are supported. Maxima and R are also powerful stand-alone programs that can be installed on most operating systems. If you are interested in writing Maxima or R problems, it is a good idea to install a copy on your operating system to access help, learn syntax, and test your expected responses outside the LON-CAPA environment. See <http://maxima.sourceforge.net/> or <http://www.r-project.org/>.

LON-CAPA will accept two pre-named arrays inside the answerblock for the computer algebra system: RESPONSE and LONCAPALIST. RESPONSE contains the student input by comma-separated entities, for example, if “3,42,17” is entered by the student, RESPONSE[2] would be 42. LONCAPALIST is built from the arguments passed in an array args which is assigned a array value from the script.

The **answer** tag contains the Maxima command (and syntax) that are passed to Maxima after the RESPONSE and LONCAPALIST values are substituted. (See example below). The **answerdisplay** variable contains what is displayed when the problem is in “Show Answer” mode. You will want to include this field so that the students can see the correct answer after the “Show Answer Date” configured when the problem is assigned in the course space. Also note the description in the **postanswerdate** tag that is displayed after the answer date.

The following example illustrates this.

```
<problem>
  <script type="loncapa/perl">
$a1 = random(-6,6,4);
$a2 = random(-6,6,4);
$n1 = random(3,11,2);
$n2 = random(2,10,2);
```

```

$function = "$a1*cos($n1*x)+$a2*sin($n2*x)";
# reformat next two lines as single line if you copy/paste into a script
$example=&xmlparse('An example would be
<m eval="on">$(sin($n1\cdot x)+cos($n2\cdot x))/\sqrt{2}$</m>');
  </script>

<startouttext />
  Give an example of a function
  <ol>
    <li>
      which is orthogonal to <algebra>$function</algebra> with respect to the
      scalar product
      <m>
        \[<g \mid h> =
          \frac{1}{\pi} \int_{-\pi}^{\pi} dx \, g(x) \cdot h(x)\]
      </m>
    </li>
    <li>
      whose norm is 1.
    </li>
  </ol>
<endouttext />

<mathresponse answerdisplay="$example" cas="maxima" args="$function">
  <answer>
    overlap:integrate((RESPONSE[1])*(LONCAPALIST[1]),x,-%pi,%pi)/%pi;
    norm:integrate((RESPONSE[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
    is(overlap=0 and norm=1);
  </answer>
  <textline readonly="no" size="50" />
  <hintgroup showoncorrect="no">
    <mathhint name="ortho" args="$function" cas="maxima">
      <answer>
        overlap: integrate((LONCAPALIST[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
        is(not overlap = 0);
      </answer>
    </mathhint>
    <mathhint name="norm" args="$function" cas="maxima">
      <answer>
        norm: integrate((RESPONSE[1])*(RESPONSE[1]),x,-%pi,%pi)/%pi;
        is(not norm = 1);
      </answer>
    </mathhint>
    <hintpart on="norm">
      <startouttext />

```

The function you have provided does not have a norm of one.

```

        <endouttext />
    </hintpart>
    <hintpart on="ortho">
        <startouttext />
The function you have provided is not orthogonal.
        <endouttext />
    </hintpart>
</hintgroup>
</mathresponse>

```

```

<postanswerdate>
    <startouttext />
    <p>
Note that with respect to the above norm, <m>$$\cos(nx)$$</m> is perpendicular
to <m>$$\sin(nx)$$</m> and perpendicular to <m>$$\cos(mx)$$</m> for
<m>$$n \neq m$$</m>.
    </p>
    <endouttext />
</postanswerdate>
</problem>

```

7.12 Custom Response Problems

Custom Response is a way to have a problem graded based on an algorithm. The use of this response type is generally discouraged, since the responses will not be analyzable by the LON-CAPA statistics tools.

For a single textfield, the student's answer will be in a variable `$submission`. If the Custom Response has multiple textfields, the answers will be in an array reference, and can be accessed as `$$submission[0]`, `$$submission[1]`, etc.

The student answer needs to be evaluated by Perl code inside the `< answer>`-tag. Custom Response needs to include an algorithm that determines and returns a standard LON-CAPA response. The most common LON-CAPA responses are:

- EXACT_ANS: return if solved exactly correctly
- APPROX_ANS: return if solved approximately
- INCORRECT: return if not correct, uses up a try
- ASSIGNED_SCORE: partial credit (also return the credit factor, e.g. `return(ASSIGNED_SCORE,0.3);`)
- SIG_FAIL, NO_UNIT, EXTRA_ANSWER, MISSING_ANSWER, BAD_FORMULA, WANTED_NUMERIC, WRONG_FORMAT: return if not correct for different reasons, does not use up a try

The `answerdisplay` is shown instead of the student response in 'show answer' mode after the answer date. The following example illustrates this:

```
<problem>
<startouttext />Accept an answer of around 90 or -90<endouttext />
  <customresponse answerdisplay="something near 90 or -90">
    <answer type="loncapa/perl">
# This examples uses perl 'regular expressions' for string evaluation.
# Consult a perl reference for help understanding the regular expressions.
# We do not want a vector
if ($submission=~/\,/){ return 'EXTRA_ANSWER'; }
# Need a numerical answer here
if ($submission!~/^[d\.\-e]+$/i){ return 'WANTED_NUMERIC'; }
$difference=abs(90-abs($submission));
if ($difference==0){ return 'EXACT_ANS'; }
if ($difference < 0.1){ return 'APPROX_ANS'; }
return 'INCORRECT';</answer>
    <textline readonly="no" />
  </customresponse>
</problem>
```

Full list of possible return codes:

- EXACT_ANS: student is exactly correct
- APPROX_ANS: student is approximately correct
- NO_RESPONSE: student submitted no response
- MISSING_ANSWER: student submitted some but not all parts of a response
- EXTRA_ANSWER: student submitted a vector of values when a scalar was expected
- WANTED_NUMERIC: expected a numeric answer and didn't get one
- SIG_FAIL: incorrect number of Significant Figures
- UNIT_FAIL: incorrect unit
- UNIT_NOTNEEDED: submitted a unit when one shouldn't
- UNIT_INVALID_INSTRUCTOR: the unit provided by the author of the problem is unparsable
- UNIT_INVALID_STUDENT: the unit provided by the student is unparasable
- UNIT_IRRECONCIBLE: the unit from the student and the instructor are of different types
- NO_UNIT: needed a unit but none was submitted

- **BAD_FORMULA**: syntax error in submitted formula
- **WRONG_FORMAT**: student submission did not have the expected format
- **INCORRECT**: answer was wrong
- **SUBMITTED**: submission wasn't graded
- **DRAFT**: submission only stored
- **MISORDERED_RANK**: student submitted a poorly order rank response
- **ERROR**: unable to get a grade
- **ASSIGNED_SCORE**: partial credit; the customresponse needs to return the award followed by the partial credit factor
- **TOO_LONG**: answer submission was deemed too long
- **INVALID_FILETYPE**: student tried to upload a file that was of an extension that was not specifically allowed
- **EXCESS_FILESIZE**: student uploaded file(s) with a combined size that exceeded the amount allowed
- **COMMA_FAIL**: answer requires the use of comma grouping and it wasn't provided or was incorrect

8 Authoring Dynamically Generated Plots

The **gnuplot** LON-CAPA tag allows an author to design a plot which will be created programmatically at the time when it is requested for display by a student. This is intended for use in homework problems where a distinct plot should be rendered for each student. It can be used in conjunction with a **script** tag to generate curve data for random plots. If you are using static data a dynamically generated plot is not appropriate because of the overhead associated with generating the plot. In that case, use a picture (see “Adding Picture” section 5.3.2).

The easiest way to create a template for a **gnuplot** tag is to use the colorful problem editor, and select **GnuPlot** from the **Insert** dropdown menu. Thereafter the boxes may be filled using this documentation to assist you. By default, only the parent **gnuplot** tag and the child **curve** tag are created by when the plot is inserted into a problem using the colorful editor. Other tags can be added to decorate and polish your plot using the **Insert**: dropdown provided by the colorful editor.

The following **gnuplot** parameters may be set:

- **brief description of the plot** This text is used as the **alt** value of the **img** tag used to display the plot on a web page.

- **background color of image** (xxxxxx) See the “Color Selection” section 8.4 for help on specifying colors.
- **foreground color of image** (xxxxxx) See the “Color Selection” section 8.4 for help on specifying colors.
- **height of image** (pixels)
- **width of image** (pixels)
- **Size of font to use** “small”, “medium”, or “large”. The font used for any text on the plot is set with this tag.
- **Transparent image** “Yes” or “No”. If the image is transparent the background color will be ignored.
- **Display grid** “Yes” or “No”.
- **Number of samples for non-data plots** If a **function** tag is used (see “Plotting Functions” section 8.1.2) to specify the **curve** (see “Specifying Curves to Plot” section 8.1), this indicates the number of sample points to use.
- **Draw border around plot** “Yes” or “No”
- **alignment for image in html** “Left”, “Center”, or “Right”. This is the value used for the **align** parameter in the **img** tag which embeds the plot in the problem.
- **Width of plot when printed (mm)** The width in mm of the plot when it is printed. The default is approximately one half of a U.S. letter size page, 93 mm.
- **Font size to use in TeX output (pts)** The size in points of text on the graph when it is printed out.
- **Plot type** “Cartesian” or “Polar”.
- **margin width (pts)** The left, right, top, or bottom margin width measured in points.
- **Size of major tic marks** The size of the larger tic marks on the plot border or axes, measured in graph units.
- **Size of minor tic marks** The size of the smaller tic marks on the plot border or axes, measured in graph units.

The parent **gnuplot** tag allows use of the following child tags (only the **curve** tag is added automatically when inserting with the colorful editor):

- **curve** (see “Specifying Curves to Plot” section 8.1)
- **title**, **xlabel**, and **ylabel** (see “Plot Labels and Key” section 8.2)
- **key** (see “Plot Labels and Key” section 8.2)

- **label** (see “Plot Labels and Key” section 8.2)
- **axes** (see “Plot Axes Details” section 8.3)
- **tics** (see “Plot Axes Details” section 8.3)

8.1 Specifying Curves to Plot

The **curve** tag is where you set the data to be plotted by gnuplot.

The following parameters may be present in the **curve** tag and allow access to most of the formatting parameters for a curve in gnuplot.

- **color**
The color of the curve on the plot. See “Selecting Colors” section 8.4.
- **name**
If a key is present, the name of the curve will appear with a sample of its line type.
- **linestyle**
See “Line Styles” section 8.1.3 for more information about the available line styles and their data requirements.
- **linetype**
The type of line. Current options include 'solid' and 'dashed'. At this time, all dashed lines draw with line width of '1' in web output. This parameter may not apply to all linestyles.
- **linewidth**
The thickness of the line drawn by plotting engine. This parameter may not apply to all linestyles.
- **pointtype**
This parameter may not apply to all linestyles.
- **pointsize**
This parameter may not apply to all linestyles. The size of the points, in pixels, present on the line. Some point types are not affected by this parameter.
- **limit**
This parameter controls the point to fill for filled curves.
- **arrowhead**
For vector plots, controls where in the vector the arrow head(s) appear
- **arrowstyle**
For vector plots controls the fill style of the arrow.

- **arrowlength**

For vector plots determines the distance between the vector line end and the tip of the arrow.

- **arrowangle**

For vector plots, determines the angle the arrow branches make with the vector line.

- **arrowbackangle**

For vector plots, determines the angle the arrow lines that return to the main line from the branches make with the arrow branches.

There are two ways of entering the information to be plotted, which are accessed using the subtags of **curve**, **data** (see “Plotting Data Points” section 8.1.1) and **function** (see “Plotting Functions” section 8.1.2).

8.1.1 Plotting Data Points

The **data** tag is used to specify the values plotted in the **gnuplot** tag (see “Authoring Dynamically Generated Plots” section 8). The **data** tag is only used in the **Curve** tag (see “Specifying Curves to Plot” section 8.1).

The data must be either a perl array, **@X**, or a comma separated list, such as “0.5,0.9,1.5,2.4” (without quotes). ‘NaN’ is a valid value. Note the the “Y” values are entered in a separate array.

The function and number of **data** tags required varies based on the line style (see “Data and Line Styles” section 8.1.3) chosen for the curve. Some linestyles require extra arrays for supplemental information. In all cases the first **data** tag will hold the “X” values and the second will hold the “Y” values.

All of the data sets in the **data** tag must have the same number of elements.

8.1.2 Plotting Functions

The **function** tag allows you to specify the curve to be plotted as a formula, instead of numerical data.

The function must be a mathematical expression. Use the independent variable “x” for cartesian plots and “t” for polar plots. Implicit multiplication is not accepted by Gnuplot. The following are examples of valid functions and invalid functions:

- **sin(x)**
- **sin(2*x)**
- **sin(x**2)**
- **exp(x)**
- **3*x**x**

- `exp(sin(2*x))`
- `sinh(x)`
- `sin(t)*cos(t)` (*polar plot only*)

8.1.3 Data and Line Styles

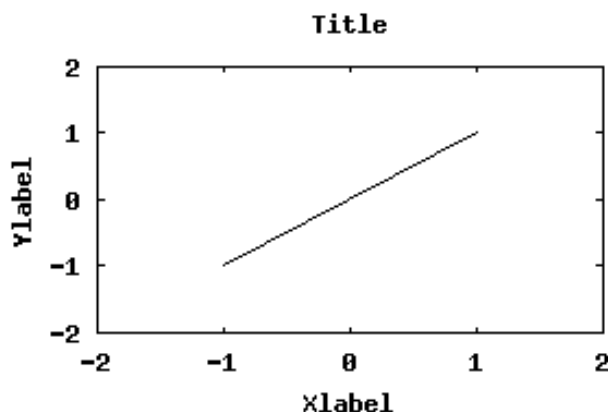
Unless otherwise noted the linestyles require only 2 data sets, X and Y.

- **lines** Connect adjacent points with straight line segments.
- **points** Display a small marker at each point.
- **linespoints** Draw both **lines** and **points**.
Draws a small symbol at each point and then connects adjacent points with straight line segments.
- **dots** Place a tiny dots on the given points.
- **steps** Connect points with horizontal lines.
This style connects consecutive points with two line segments: the first from (x1,y1) to (x2,y1) and the second from (x2,y1) to (x2,y2).
- **fsteps** Connect data with horizontal lines.
This style connects consecutive points with two line segments: the first from (x1,y1) to (x1,y2) and the second from (x1,y2) to (x2,y2).
- **histeps** Plot as histogram.
Y-values are assumed to be centered at the x-values; the point at x1 is represented as a horizontal line from $((x0+x1)/2, y1)$ to $((x1+x2)/2, y1)$. The lines representing the end points are extended so that the step is centered on at x. Adjacent points are connected by a vertical line at their average x, that is, from $((x1+x2)/2, y1)$ to $((x1+x2)/2, y2)$.
- **errorbars** Same as yerrorbars.
- **xerrorbars** Draw horizontal error bars around the points.
Requires 3 or 4 data sets. Either X, Y, Xdelta or X, Y, Xlower, Xupper. Xdelta is a change relative to the given X value. The Xlower and Xupper values are absolute grid coordinates of the upper and lower values to indicated with error bars.
- **yerrorbars** Draw vertical error bars around the points.
Requires 3 or 4 data sets. Either X, Y, Ydelta or X, Y, Ylower, Yupper. Ydelta is a change relative to the given Y value. The Ylower and Yupper values are the grid coordinates of the upper and lower values to indicate with error bars.

- **xyerrorbars** Draw both vertical and horizontal error bars around the points.
Requires 4 or 6 data sets. Either X, Y, Xdelta, Ydelta or X, Y, Xlower, Xupper, Ylower, Yupper. Xdelta and Ydelta are relative to the given coordinates. Xlower, Xupper, Ylower, and Yupper are the grid coordinates of the upper and lower values to indicate with the error bars.
- **boxes** Draw a box from the X-axis to the Y-value given.
Requires either 2 or 3 data sets. Either X, Y or X, Y, Xwidth. In the first case the boxes will be drawn next to eachother. In the latter case Xwidth indicates the horizontal width of the box for the given coordinate.
- **vector** Draws a vector field based on the given data.
Requires 4 data sets, X, Y, Xdelta, and Ydelta. The ‘vector’ style draws a vector from (X,Y) to (X+Xdelta,Y+Ydelta). It also draws a small arrowhead at the end of the vector. May not be fully supported by gnuplot.

8.2 Plot Labels and Key

Three of the more basic tags are **title**, **xlabel**, and **ylabel**. Their size and color depend on the values chosen for the font size and graph foreground color specified in the **gnuplot** tag. The figure below shows the locations of the various labels.



The **key** tag causes a key to be drawn on the plot when it is generated. The key will contain an entry for each **curve** which has a name (see “Specifying Curves to Plot” section 8.1).

The key is the color of the foreground of the plot, specified in the **gnuplot** tag (see “Authoring Dynamically Generated Plots” section 8).

The **label** tag allows the author to place text at any position on the plot. There may be many **label** tags on one plot and all the labels which fall within the plot will show. The color used will be to foreground color of the plot and the font will be the size specified for the plot, both of which are set in the **gnuplot** tag (see “Authoring Dynamically Generated Plots” section 8).

- justification of the label text on the plot “left”, “right”, or “center”.

- rotation of label (degrees)
- x position of label (graph coordinates)
- y position of label (graph coordinates)

The text to be placed on the plot must be entered as well.

8.3 Plot Axes Details

The **Plot Axes** tag allows you to specify the domain and range of the data to display. It is closely tied with the **Plot Ticks** 8.3 tags, which specify where the gridlines are drawn on the plot. The **Plot Axes** tag sets the following parameters:

The color of grid lines

If the “Display Grid” parameter of the Gnuplot tag is set to yes, the grid will be displayed in the specified color. Hexadecimal notation is used to specify the color (see Color Selection section 8.4).

The view of the graph shown

The viewing rectangle of the graph is set with the following parameters:

- minimum x-value
- maximum x-value
- minimum y-value
- maximum y-value

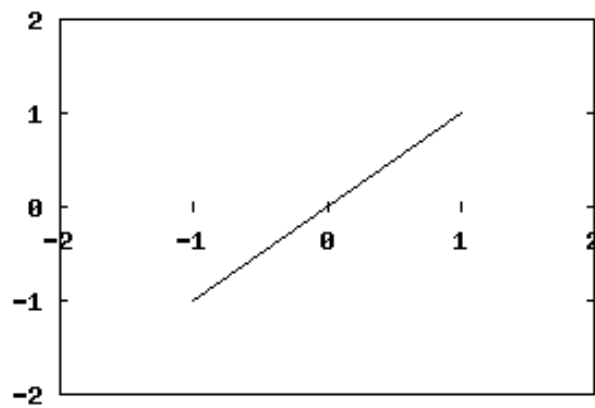
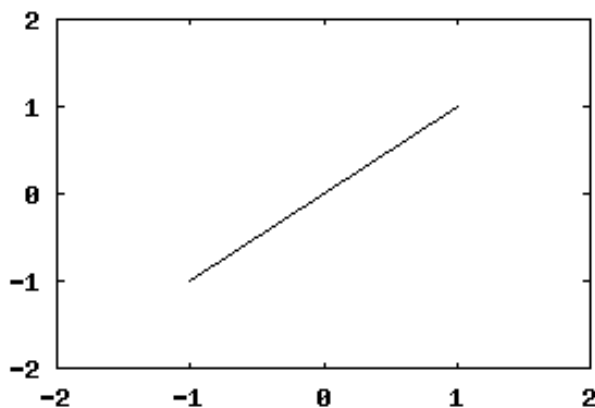
See also the Plot Ticks section 8.3 and the general Gnuplot help section 8.

The **xtics** and **ytics** tags can be inserted by selecting the **Plot ticks** item from the insert selection list of the **gnuplot** tag.

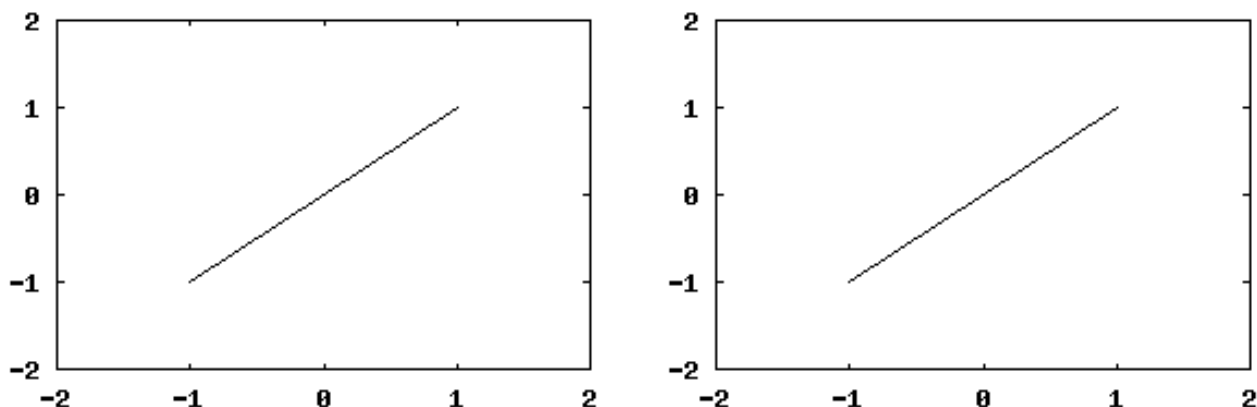
The **xtics** and **ytics** tags have identical structure and the description presented here applies to both.

The ticks tags allow specification of the following attributes:

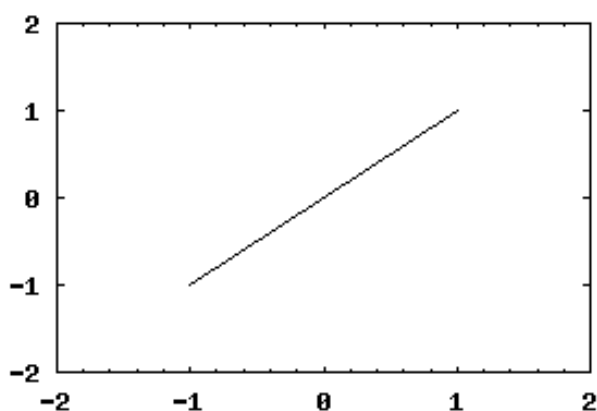
- Location of major tic marks “Border” or “Axis”. Tic marks can be placed on the border or on the axes. The images below illustrate the effects of each of these options.



- **Mirror tics on opposite axis?** “Yes” or “No”. If the **location of tic marks** is set to “border” this parameter determines if they are shown on both the top and bottom or right and left sides of the graph. The “mirror” tic marks are unlabelled.



- **Start major tics at**
The point in graph coordinates which to start making major tics. This may be less than or greater than the lower limit for the axis.
- **Place a major tic every**
The span, in graph coordinates, between each major tic mark.
- **Stop major tics at**
This may be less than or greater than the upper limit for the axis.
- **Number of minor tics between major tic marks**
The number of subdivisions to make of the span between major tic marks. Using a value of “10” leads to 9 minor tic marks. The example below uses a value of “5” to produce 4 tic marks.



- **rotate**
For output devices that support it, the **rotate=‘1’** attribute rotates the tic label. This is most useful with large labels defined by the **tic** tag described below.

In addition to specifying regular tic intervals via the attributes of the **xtics** and **ytics** you can specify arbitrary tic locations by enclosing **tic** tags within the body of these tags. Each **tic** tag requires the **location** attribute which specifies the location of a tic on the axis. The body of the tag contains the label of that tic.

The **xtic** or **yctic** **rotate** attribute can rotate the label text if the output device supports text rotation. If **tic** tags are embedded within the **xtic** or **yctic** tags attributes of those tags that specify tic boundaries and interval are ignored. Here's an example of an X axis tic specification in months of the year:

```
...
<xtics rotate='on'>
  <tic location='1'>January</tic>
  <tic location='2'>February</tic>
  <tic location='3'>March</tic>
  <tic location='4'>April</tic>
  <tic location='5'>May</tic>
  <tic location='6'>June</tic>
  <tic location='7'>July</tic>
  <tic location='8'>August</tic>
  <tic location='9'>September</tic>
  <tic location='10'>October</tic>
  <tic location='11'>November</tic>
  <tic location='12'>December</tic>
</xtics>
<ytics end="6.0" location="border" start="-6.0"
  increment="1.0" mirror="on" />
```

Note that tic locations are completely under your control and do not even have to be at regular intervals on the axis if that better suits your needs.

8.4 Color Selection

The default colors are a white background (xffffff) with black (x000000) foreground, gridlines, and curve.

- Background color is an attribute of the **gnuplot** tag (see “Authoring Dynamically Generated Plots” section 8). This controls the color of the plot image. The default is white (xffffff).
- Foreground color is an attribute of the **gnuplot** tag (see “Authoring Dynamically Generated Plots” section 8). This controls the color of the border.
- Gridline color is an attribute of the **axis** tag (see “Plot Axes Details” section 8.3).
- Curve color is an attribute of the **curve** tag (see “Specifying Curves to Plot” section 8.1). This is the color of the curve function or data points. Different curves can be given different colors.

9 Authoring Function Plot Response Problems

9.1 Introduction

Creating a **Function Plot Response** problem involves several steps. This section introduces the major settings and subsequent sections cover other details.

“Label x-axis” and “Label y-axis” - Enter the label and/or units for the axes on the graph. Leaving them blank will result in no axis labels.

“Minimum x-value”, “Maximum x-value”, “Minimum y-value”, and “Maximum y-value” - Entering values here will set the value of the graph at the left, right, bottom, and top edges (respectively). Default is -10, 10, -10, and 10 respectively.

“x-axis visible” and “y-axis visible” - This allows for each axis to be turned on or off.

“Grid visible” - This determines whether or not the grid is on the graph.

“Background plot(s) for answer” - This is a green curve the computer will display once the correct answer has been submitted. It is static, and can be given as a piecewise function. Since some problems will have multiple correct answers, this necessarily will only be a possible answer. Only the left hand side of the equation is necessary. For example, entering $x + 2$ will display the line $y = x + 2$. The syntax must be syntax recognized by GeoGebra. To test syntax for Geogebra directly, visit <http://www.geogebra.org/webstart/geogebra.html>.

The tag should include the following additional subsections

- Response Elements
- Rule Set

9.2 Elements

Which Elements you use depends on the type of problem you want to make.

- Background Plot - This places a static curve on the graph of your choosing. It can be labeled, moveable or fixed, and any color desired. Only the right hand side of the function you want displayed is necessary. For example, entering $x + 2$ will display the line $y = x + 2$. The syntax must be syntax recognized by GeoGebra. To test syntax for Geogebra directly, visit <http://www.geogebra.org/webstart/geogebra.html>.
- Spline - At least one spline is necessary for a graph problem. These splines are what will be adjusted and analyzed to solve the problem.
- Object - This places a point in the applet. Generally intended to be used with Vectors to create problems involving Free-Body Diagrams or any other points that vectors (or arrows) connect to and from.
- Vector - This creates a vector (or arrow) in the applet. Generally intended to be used with Objects to create problems involving Free-Body Diagrams or to establish connections between Objects.

Splines

“Index” - This is the label assigned to the spline. In general, it’s simplest just to label them A, B, C etc.

“Order” - This determines the number of Control Points on the spline. For example, selecting ‘3’ means there will be 3 points on the spline that can be moved, as well as 3 points off the spline that will control the slope.

“Initial x-value” and “Initial y-value” - These values determine where the left most Control Point will be.

“Scale x” - This determines the right most location of the Control Points (on the spline). To figure out where this point will be, add ‘Initial x-value’ to ‘Scale x’.

“Scale y” - This determines the distance (in the y-direction) between the Control Points on the spline, and the ones that control the slope.

Vectors

- “Label on Plot” - Determines the name of the vector, as well as the name that will be visible in the problem. This value **MUST** be capitalized and cannot include spaces or most symbols. To be safe, stick with letters and numbers.
- “Tail x”, “Tail y”, “Tip x”, and “Tip y” - Determines where the vector’s tail and tip will be placed in the applet when the problem is first loaded. Vectors are moveable by the students, and will not revert to these values after the student has submitted an answer.

Objects

- “Label on Plot” - Determines the name of the object, as well as the name that will be visible in the problem. This value **MUST** be capitalized and cannot include spaces or most symbols. To be safe, stick with letters and numbers.
- “x” and “y” - Determines where the object will be placed in the applet. The object is not moveable by the students, so whatever value is chosen here is fixed.

Background Plots

“Function” - Enter the function here. An equals sign is not necessary. Just give the right hand side of the function. LON-CAPA variables are usable as well to allow individualized problems for each student. The syntax must be syntax recognized by GeoGebra. To test syntax for Geogebra directly, visit <http://www.geogebra.org/webstart/geogebra.html> .

“Initial x-value” and “Final x-value” - The function does not need to span the entire graph. It is possible to create a piecewise function by defining the x-values over which each function is to be shown.

“Label on Plot” - To label the background function (for example, as “parabola”), enter it here.

“Color” - To change the color of the background function, enter the hex code for that color here. The default is 000000 (black). It is recommended to choose a color other than green, since it is easily confused with being the answer.

9.3 Rules

This is where the rules are defined. These rules will determine whether or not an entered answer is correct or not. If there are no rules, any answer will be deemed correct. If there is more than one rule, when an answer is submitted, the server will analyze them in order until one of them is broken (of course, if it's a correct answer, it will go through all of them and return a green box). In such an event, any subsequent rules will be ignored. If conditional hints related to these rules are added, only the first broken rule's hint will be shown, even if all rules are broken.

- “Graph Rule” - Used to evaluate graph problems.
- “Vector Rule” - Used to test whether vectors are in the right place, pointed in the right direction, and have the correct length.
- “Vector Sum Rule” - Used to test the sum of a set of vectors.
- “Custom Rule” - Used to create rules that the other rules can't do. (Advanced)

Graph Rule

Overview - This box is used to create a rule that determines whether or not a submitted graph is correct. In general, it takes the form of testing the function, its integral, or its first or second derivative over a given set of x-values. The test can be to see if it equals, is greater than, or less than a specified value. Anywhere a number is needed, a variable can also be used. (Skip to the bottom for examples of rules.)

- “Index/Name” - This is an internal label for the rule. Something must be entered here, and it must be different for each rule. This same value will be used to add a conditional hint.
- “Function” - This determines what the server will be testing. For instance, choose 'First derivative' causes the server to evaluate the derivative of the entered answer over the given domain.
- “Initial x-value” and “Initial x-value label” - A value must be entered for one of these. Either choose a numerical value for x (the first option), or choose the beginning of the submitted answer, the end, or a previously chosen named point (see below).
- “Final x-value” and “Final x-value label” - This determines the end of the domain over which the rule examines. To test only a single point (the initial value), leave these blank. If a label is entered, such as 'positive', the point at which the rule fails will be given this special label. This label can then be used in subsequent rules as an 'Initial x-value label'.

- “Minimum length for range” - This tests that the difference between the initial and final x-values are at least a certain length apart. This is only useful if there is at least one label.
- “Maximum length for range” - This tests that the difference between the initial and final x-values are at most a certain length apart. This is only useful if there is at least one label.
- “Relationship” - The heart of the rule. This choice determines whether the chosen ‘Function’ is greater than, less than, equal to, etc. a certain ‘Value’.
- “Value” - Enter the number you wish to compare to. It is also possible to choose ‘not defined’, in the event the answer should not have a value for the given domain. Within the value argument, the function itself can be evaluated using `&fpr_f()`, its derivative using `&fpr_dfdx()`, and its second derivative using `&fpr_d2fdx2()`. This allows for a comparison of two points on the graph. The value of a previously defined label can be retrieved using the function `&fpr_val()`, e.g., `&fpr_val(‘positive’)`. Previous defined values from script blocks can also be retrieved as normal variables, e.g., `$x`.
- “Percent error” - This allows for a margin of error in the y-direction. For instance, if the rule requires that the derivative be equal to 5, the server will accept values close enough to 5 that are within the percent error defined here. Note: Choosing 10% would not mean that the answer is correct as long as it is within the range 4.5-5.5. Instead, the percent corresponds to the total size of the graph. For the function itself, the ‘percent error’ is multiplied by the $y_{\max}-y_{\min}$; for the first derivative, it’s multiplied by $(y_{\max}-y_{\min})/(x_{\max}-x_{\min})$; for the second derivative, it’s multiplied by $(y_{\max}-y_{\min})/(x_{\max}-x_{\min})^2$; and for the integral, it’s multiplied by $(y_{\max}-y_{\min})*(x_{\max}-x_{\min})$.

The figure below shows some examples of rules.

Function Plot Rule Set Delete? Insert: [Function Plot Rules](#)

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: 0 Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: 0 Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: 7 Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional): notzero

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: 0 Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label: notzero

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: 0 Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: &fpr_f(\$time2) Percent error:

Insert:

Function Plot Graph Rule Delete? [Evaluation Rules](#)

Index/Name: Function:

Initial x-value: Initial x-value label:

Final x-value (optional): Final x-value label (optional):

Minimum length for range (optional): Maximum length for range (optional):

Relationship: Value: &fpr_f(&fpr_val("notzero")) Percent error:

Insert:

1. Checks that the derivative at $x = 3$ is negative.

2. Checks that the second derivative at $x = \text{\$time1}$ is 0.
3. Checks that the value of the function > 7 from “Start of Graph” to $x = 5$.
4. Checks that the function is 0 from $x = 0$ until the function is no longer 0, labeling this new point ‘notzero’.
5. Checks that the first derivative is positive between the point ‘notzero’ and the end of the graph.
6. Checks to see that the value of the function at $x = 4$ is the same as the value of the function at $\text{\$time2}$. The Value here is $\&\text{fpr_f}(\text{\$time2})$.
7. Checks that the value of the function at point ‘notzero’ is equal to the value of the function at $x = \text{\$time3}$. The Value here is $\&\text{fpr_f}(\&\text{fpr_val}(\text{‘notzero’}))$.

Vector Rule

- “Index/Name” - This is an internal label for the rule. Something must be entered here, and it must be different for each rule. This same value will be used to add a conditional hint.
- “Vector” - The name of one of the vectors in the list of Elements above. Specifically, the one you want to test.
- “Attached to object” - Enter the object(s) this vector should be attached to. For more than one object, separate them by commas. If the vector should not be attached to any object, leave this blank. In this case, an object is considered attached if its tail OR its tip is in the vicinity of the object.
- “Not attached to object” - Enter the object(s) that this vector should be not be near. For more than one object, separate them by commas. Particularly useful for distractor vectors.
- “Tail/Tip (not) attached to object” - The same as the previous two statements, but specific to only the Tail/Tip of the vector.
- “Length” - How long the vector should be.
- “Absolute error length” - How accurate the length must be to get the answer correct.
- “Angle” - What direction should the vector point. All values are measured in degrees, counterclockwise starting at the positive x-axis. Values must be $0 \leq \theta < 360$
- “Absolute error angle” - How accurate the angle must be to get the answer correct.

Vector Sum Rule

- “Index/Name” - This is an internal label for the rule. Something must be entered here, and it must be different for each rule. This same value will be used to add a conditional hint.
- “Comma-separated list of vectors” - List all of the vectors that should be added up to be tested.
- “Length” - How long the sum of these vectors should be.
- “Absolute error length” - How accurate the length must be to get the answer correct.
- “Angle” - What direction should the sum of these vectors point. All values are measured in degrees, counterclockwise starting at the positive x-axis. Values must be $0 \leq \theta < 360$
- “Absolute error angle” - How accurate the angle must be to get the answer correct.

Custom Rule

Used to create rules that aren't options using the other rules. The coding is done in Perl and follows Perl syntax. Any variable written inside this rule will be recognized as normal and any evaluation function can be used as well.

Available evaluation functions:

1. `&fpr_val("label")`
2. `&fpr_f($x)`
3. `&fpr_dfdx($x)`
4. `&fpr_d2fdx2($x)`
5. `($xs,$xe,$ys,$ye)=&fpr_vectorcoords("Name")`
6. `($x,$y)=&fpr_objectcoords("Name")`
7. `&fpr_vectorlength("Name")`
8. `&fpr_vectorangle("Name")`

Returning 1 is correct, while returning 0 is incorrect.

Example comparing the lengths of two vectors:

```
if (&fpr_vectorlength('Normal') < &fpr_vectorlength('Gravity'))
{return 1;}
else
{return 0;}
```

10 Using Libraries

Library (.library) files can hold an assortment of content. Library content is loaded into a problem statement by using an `<import>` tag. For more information about the import tag, see section 15.8.

10.1 Authoring Library Files

A LON-CAPA .library file can contain just a script block, or just response items, or both. A LON-CAPA problem can import as many published library files as desired. A .library file always starts with a `<library>` tag, and always ends with a `</library>` tag.

Storing entire scripts

Entire scripts can be stored in a library file. The entire script can then be imported into a problem file.

Library file:

```
<library>
<script type="loncapa/perl">
@alpha=('A','B','C','D',);
$seed=&random(1,1000000,1);
@alpha=&random_permutation($seed,@alpha); #scramble order
$letter = $alpha[0]; #select first element
</script>
</library>
```

Problem file:

```
<problem>
<import id="15">randomletter.library</import>
<startouttext />The random letter is $letter.<endouttext />
<!-- other problem tags could go here -->
</problem>
```

Storing a portion of a script

A portion of a script, such as a large data array can be stored in a library file.

Library file:

```
<library>
<script type="loncapa/perl">
@alpha=('A','B','C','D',);
$seed=&random(1,1000000,1);
@alpha=&random_permutation($seed,@alpha); #scramble order
</script>
</library>
```

Problem file: (note the `<script>` tag is repeated and other script calculations can be done using variables from the library file.)

```
<problem>
<import id="15">randomletter.library</import>
<script type="loncapa/perl">
$letter = $alpha[0];
</script>
<startouttext />The random letter is $letter.<endouttext />
<!-- other problem tags could go here. -->
</problem>
```

Storing a subroutine

Another use of a .library file is to define a subroutine which you plan to call in a number of instances, e.g., (see notes below about browsing libraries in the repository to see the contents of this subroutine)

/res/msu/raeburn/cleaneq.library

Here is some example XML problem code which makes a call to the `&cleaneq()` routine defined in the library file, passing some arguments: `$eq,'x','y','z'` in the call to the routine.

```
<problem>
<import id="15">/res/msu/raeburn/cleaneq.library</import>

<script type="loncapa/perl">
$eq = "1x + 0y +-7z --3";
$eq2 = &cleaneq($eq,'x','y','z');</script>
<startouttext />Here is an example equation:<br />
Without cleaneq: $eq<br />
With cleaneq: $eq2<endouttext />
</problem>
```

Assigning random problems using libraries Libraries can be used to store alternative parts of problems which are selected with the `<randomlist>` tag. The .library file hold the all content that would normally appear inside the `<part>` tag.

```
<part id="11">
<randomlist show="1">
<import id="12">sample1.library< /import>
<import id="13">sample2.library< /import>
<import id="14">sample3.library< /import>
< /randomlist>
< /part>
```

```

<part id="15">
<randomlist show="1">
<import id="16">sample4.library< /import>
<import id="17">sample5.library< /import>
<import id="18">sample6.library< /import>
< /randomlist>
< /part>

```

Note: when using `<randomlist>` as shown above, all students will work every part of the problem, but the actual problem statements will be different. Another option is to wrap multiple `<part>` tags but then not all students will work all parts unless the value of ‘show’ equals the total parts wrapped. For more information see section 15.8.

Viewing the text contents of a library script block

If you click on a .library file when browsing the shared content repository, and the .library file contains just a script block, then nothing will be displayed in the pop-up window.

The code is viewable if the author has enabled access to the source XML when publishing a .library item that is pure script block. If that is done, then when a user checks the “Source Available” checkbox when browsing the shared content pool, a link will be displayed for items with available source code. Clicking the “Source Code” link for any such items will open a pop-up which displays the content of the library file. It is good practice to enable access to the source code when publishing any library that will be shared. Otherwise, users cannot see it.

Viewing variables from a library script during testing

When viewing a problem in the problem testing mode of Authoring Space, you will see a separate Script Vars link at the bottom of the testing area for each script block (either a block included directly within the file, or a block included within a library file imported into a problem). By clicking the respective link, you can view variable values from the respective script.

Accessing submissions from a problem part loaded from a library

When `*response` items (e.g., `*response` is a wildcard such as `optionresponse`, `stringresponse`, `numericalresponse`, etc.) are defined in a .library file, this results in an extra id item in the identifier required in `&EXT()` functions, e.g., if a problem contains two parts with ids of a and b respectively, and the id of the `<import>` for the .library is 15, and the `*response` items have ids of 11 and 12 respectively, the most recent submissions could be retrieved with the following `&EXT()` calls.

```

&EXT('user.resource.resource.a.15_11.submission');
&EXT('user.resource.resource.b.15_12.submission');

```

11 Authoring Adaptive/Conditional Hints

Hints are placed within `<hintgroup></hintgroup>` tags. The first part of the hint is the condition, which includes a specification of the foil(s) and foil answer(s) required to trigger

the hint. The answers specified in the hint condition are compared with the user's submission, and if the condition is met, the hint action included in the conditional hint block will be executed (for example this could be the display of a block of text). You can set multiple hint conditions for a particular problem. Hint conditions are identified by a name. The corresponding hint action includes this hint condition name in the "on" parameter. When a hint condition evaluates to true, the corresponding hint action is triggered. Besides providing hint actions within `<hintpart on="NAME"></hintpart>` tags for each named (NAME) hint condition, a hint can be designated for display if none of the conditional hints evaluate to true. The default hint is not displayed if the conditions were met for any of the conditional hints. The default hint action is included between `<hintpart on="default"></hintpart>` tags. There are five types of hint condition:

- Formula Hint condition
- Numerical Hint condition
- Option Response Hint condition
- Radiobutton Hint condition
- String Hint condition

The syntax used to describe the foil(s) and the foil answer(s) differ for the five types:

1. Formula Hint condition

The `formulahint` tag takes three parameters: answer, name, and samples. The "name" is the unique name given to the hint condition. The formula answer for which you wish to provide conditional is included in the answer parameter. The samples parameter includes the points (or range of points) over which sampling of the student's submitted answer and the formula included in the formula hint answer parameter are to be compared. The syntax is the same as used to specify sampling points in the samples parameter of the formula response tag itself. The formula submitted by the student is evaluated at the sample points for the hint and the calculated values are compared with the corresponding values determined by evaluating the "hint" answer at the same sampling points. A close correspondence between the two sets of values will trigger the hint action specified in the `<hintpart>` tag.

```
<problem>
  <script type="loncapa/perl" ># Enter computations here
    $x1 = random(2,4,1);
    $y1 = random(3,7,1);

    $x2 = random($x1+1,9,1);
    $y2 = random($y1+1,15,1);

    $m = "($y2-$y1)/($x2-$x1)";
```

```

$b = "$y1-$m*$x1";
$answer = "$m*x+$b";
$answer =~ s/\+/-/g;
$answer =~ s/-\+/-/g;

$inverted = "($x2-$x1)/($y2-$y1)";
$wrongans = "$inverted*x";
$wrongans =~ s/\+/-/g;
$wrongans =~ s/-\+/-/g;
</script>
<startouttext /><p> What is the equation of the line
which passess through ($x1,$y1) and ($x2,$y2)?</p> > y = <endouttext />
<formularesponse samples="x@-5:5#11" id="11" answer="$answer">
  <textline size="25" />
  <hintgroup>
    <formulahint samples="x@-5:5#11" answer="$wrongans" name="inversegrad">
      </formulahint>
      <hintpart on="inversegrad">
        <startouttext />You have inverted the slope in the question. Slope is
(y2-y1)/(x2 - x1) you have the slope as (x2-x1)/(y2-y1).<endouttext />
        </hintpart>
      </hintgroup>
    </formularesponse>
  </problem>

```

2. Numerical Response condition

The numericalhint tag takes four parameters: answer, name, unit and format. The “name” is the unique name given to the hint condition. The numerical answer for which you wish to provide conditional is included in the answer parameter. Student submission of that answer in combination with the “unit” parameter in the hint condition will trigger the hint action specified in the <hintpart> tag.

```

<problem> <startouttext /> A car travels 10 km in 10 min.
What is the speed of the car?<endouttext />
<numericalresponse format="1f" unit="km/hr" answer="60">
  <responseparam description="Numerical Tolerance" default="2%"
    type="tolerance" name="tol" />
  <textline />
  <hintgroup>
    <numericalhint format="1f" unit="km/min" answer="100" name="speed">
      <responseparam description="Numerical Tolerance" type="tolerance"
        default="2%" name="tol" />
    </numericalhin >
    <hintpart on="speed">

```

```

        <startouttext />You multiplied the distance by the time. Remember
        speed = distance/time<endouttext />
    </hintpart>
</hintgroup>
</numericalresponse>
</problem>

```

3. Option Response Hint condition

There are two types of option response hint conditions: one for standalone foils and one for concept groups. In both cases the option hint tag includes two parameters: answer and name for standalone foils, and concept and name for foils grouped together in a concept group. For the answer parameter, the names and submitted values for each of the foils that are being included in the hint condition are provided in a hash, i.e., in the format: ('Foil1'=>'True','Foil2'=>'False'). In the case of a conditional hint for a concept group, the format of the concept parameter is also a hash that links the name of each concept group included in the hint condition to either 'correct' or 'incorrect' - e.g., <optionhint concept="('buoyancy'=>'correct','density'=>'correct')" name="fluids" /> If 'correct' is specified for a named concept then when the conditional hint is evaluated answers for each of the foils selected by a student must be correct for the hint action to be triggered. If anything other than 'correct' is provided in the concept hash in the optionhint tag then then students answers will be compared with the set answers for the foils in the concept group and as long as at least one answer is incorrect (i.e., the concept group was not correctly answered) then the corresponding hint action will be triggered.

(a) optionresponse

```

<problem>
  <startouttext />For each of the following rock types, indicate
  whether or not the rock is a volcanic rock.<endouttext />
  <optionresponse max="10" randomize="yes">
    <foilgroup options="('Yes','No')">
      <foil location="random" value="No" name="schist">
        <startouttext />Schist<endouttext />
      </foil>
      <foil location="random" value="No" name="marble">
        <startouttext />Marble<endouttext />
      </foil>
      <foil location="random" value="Yes" name="basalt">
        <startouttext />Basalt<endouttext />
      </foil>
      <foil location="random" value="No" name="gabbro">
        <startouttext />Gabbro<endouttext />
      </foil>
      <foil location="random" value="No" name="granite">

```

```

    <startouttext />Granite<endouttext />
  </foil>
</foilgroup >
<hintgroup >
  <optionhint answer="('schist'=>'Yes','marble'=>'Yes')"
    name="metamorphic"/>
  <optionhint answer="('gabbro'=>'Yes','granite'=>'Yes')"
    name="plutonic" />
  <hintpart on="metamorphic">
    <startouttext />Schist and Marble are both examples of
      metamorphic rocks as described on page 2 of the textbook.
    <br /><br /><endouttext />
  </hintpart>
  <hintpart on="plutonic">
    <startouttext />Granite and Gabbro are both examples of
      igneous rocks that crystallized beneath the surface, i.e.,
      they are plutonic rocks.<br /><br /><endouttext />
  </hintpart>
  <hintpart on="default" >
    <startouttext />Volcanic rocks are described on page 22
      of the textbook.<endouttext />
  </hintpart>
</hintgroup>
</optionresponse>
</problem>

```

(b) optionresponse with concept groups

```

<problem>
  <startouttext />Choose the likely plate boundary type,
    where you are most likely to encounter each of the following
    geologic features or phenomena.<endouttext />
  <optionresponse max="10" randomize="yes">
    <foilgroup options="('Convergent','Divergent','Transform')" >
      <conceptgroup concept="faulting">
        <foil name="normal" value="Divergent">
          <startouttext />Normal faults<endouttext />
        </foil>
        <foil name="strike" value="Transform">
          <startouttext />Strike-slip faults<endouttext />
        </foil>
        <foil name="thrust" value="Convergent">
          <startouttext />Thrust faults<endouttext />
        </foil>
      </conceptgroup>
      <conceptgroup concept="earthquakes">

```

```

    <foil name="deep" value="Convergent">
      <startouttext / >Large Magnitude, deep and intermediate
        focus earthquakes<endouttext />
    </foil>
    <foil name="shallow" value="Transform">
      <startouttext / >Large magnitude, shallow focus earthquakes
        <endouttext />
    </foil>
    <foil name="lowmag" value="Divergent">
      <startouttext / >Low magnitude shallow focus earthquakes
        <endouttext />
    </foil>
  </conceptgroup>
  <conceptgroup concept="topography">
    <foil name="gentle" value="Divergent">
      <startouttext / >Broad area of elevated topography with a
        central rift valley.<endouttext />
    </foil>
    <foil name="linear" value="Transform">
      <startouttext / >A narrow linear fault zone with limited
        topographic expression.<endouttext />
    </foil>
    <foil name="trench" value="Convergent">
      <startouttext />A deep trench adjacent to a volcanic arc.
        <endouttext />
    </foil>
  </conceptgroup>
  <conceptgroup concept="volcanism">
    <foil name="explosive" value="Convergent">
      <startouttext />Explosive volcanism involving volatile-rich
        viscous magma.<endouttext />
    </foil>
    <foil name="fluid" value="Divergent">
      <startouttext />Non-explosive outpourings of low-viscosity
        magma.<endouttext />
    </foil>
    <foil name="nonvolcanic" value="Transform">
      <startouttext />No volcanic activity.<endouttext />
    </foil>
  </conceptgroup>
</foilgroup>
<hintgroup>
  <optionhint concept="('earthquakes' => 'incorrect')">
    name="quakes" />
  <optionhint concept="('volcanism' => 'incorrect')">

```

```

        name="volcactivity" />
<optionhint concept="('topography' => 'incorrect')"
    name="relief" />
<hintpart on="volcanism">
    <startouttext />Volcanism requires a source of magma.
    Magma can be generated by either a depression in the solidus
    caused by an influx of volatiles, or by decompression melting.
    Magma rich in volatiles tends to cause explosive volcanism.
    <endouttext />
</hintpart>
<hintpart on="quakes">
    <startouttext />Earthquakes require brittle failure, so there
    is a correlation between the depth of earthquake foci and the
    geotherm. The geotherm is depressed in subduction zones, and
    elevated at mid-ocean ridges.<endouttext />
</hintpart>
<hintpart on="relief">
    <startouttext />On a broad scale topography is an expression
    of density variation, as embodied in the concept of isostasy.
    Density variation in the earth is a function of temperature and
    composition. Cold oceanic crust entering a subduction zone is
    dense, whereas as magma is buoyant and of lower density.
    <br /><br /><endouttext />
</hintpart>
<hintpart on="default">
    <startouttext />The characteristics of plate boundaries are
    described on page 52 of the textbook.<endouttext />
</hintpart>
</hintgroup>
</optionresponse>
</problem>

```

4. Radiobutton Hint condition

The radiobutton hint tag takes two parameters: answer and name. The name is the name of the hint condition, and the answer is an array. The first element of the array will be 'foil'; the remaining elements are the names of the foils that you require to have been checked by the student for the hint to be displayed. For example, if you create a radiobutton response problem with six foils named: granite, gabbro, gneiss, shale, sandstone and schist, and you want your hint named: igneous to be displayed when either granite or basalt had been checked your radiobutton hint would be as follows:

```

<radiobuttonhint answer="('foil','granite','gabbro')"
name="igneous"></radiobuttonhint>

```

In order to trigger display of this hint you also need to create a `<hintpart></hintpart>` block that will include a textblock that contains the text of the actual hint.

```
<hintpart on="igneous">
  <startouttext />This type of rock is composed of interlocking
    crystals, a characteristic of igneous rocks.<endouttext />
</hintpart>
```

The complete radiobutton response would look as follows:

```
<problem>
  <startouttext />
  Which of the following is a sedimentary rock?
  <endouttext />

  <radiobuttonresponse max="3" randomize="yes">
    <foilgroup>
      <foil location="random" value="false" name="granite">
        <startouttext />
        Granite
        <endouttext />
      </foil>

      <foil location="random" value="false" name="gabbro">
        <startouttext />
        Gabbro
        <endouttext />
      </foil>

      <foil location="random" value="false" name="schist">
        <startouttext />
        Schist
        <endouttext />
      </foil>

      <foil location="random" value="false" name="gneiss">
        <startouttext />
        Gneiss
        <endouttext />
      </foil>

      <foil location="random" value="true" name="shale">
        <startouttext />
        Shale
```

```

        <endouttext />
    </foil>
    <foil location="random" value="true" name="sandstone">
        <startouttext />
        Sandstone
        <endouttext />
    </foil>

</foilgroup>
<hintgroup>
    <radiobuttonhint answer="('foil','granite','gabbro')"
name="igneous" ></radiobuttonhint>
    <radiobuttonhint answer="('foil','gneiss','schist')"
name="metamorphic"></radiobuttonhint>

    <hintpart on="igneous">
        <startouttext />This type of rock is composed of interlocking
        crystals, a characteristic of igneous rocks.<br /><br /><endouttext />
    </hintpart>
    <hintpart on="metamorphic">
        <startouttext />This type of rock is composed of oriented crystals, a
        characteristic of foliated metamorphic rocks.<endouttext />
    </hintpart>
</hintgroup>
</radiobuttonresponse>
</problem>

```

5. String Hint condition

The radiobutton hint tag takes two parameters: answer and name. The name is the name of the hint condition, and the answer is a text string. The type parameter allows you to choose between case sensitive, case insensitive, and case insensitive in any order. A simple example is shown below.

```

<problem>
    <startouttext />Which US state has Lansing as its capital?<endouttext />
    <stringresponse answer="Michigan" type="ci">
    <textline size="20" />
    <hintgroup>
        <stringhint answer="wisconsin" type="cs" name="wisc">
        </stringhint>
        <stringhint answer="minnesota" type="cs" name="minn">
        </stringhint>
    <hintpart on="wisc">
        <startouttext />The state capital of Wisconsin is Madison.<endouttext />
    </hintpart>
    </hintgroup>
</problem>

```

```

</hintpart>
<hintpart on="minn">
  <startouttext />The state capital of Minnesota is St. Paul.<endouttext />
</hintpart>
<hintpart on="default">
  <startouttext />The state you are looking for is also known as the
    'Great Lakes State'<endouttext />
</hintpart>
</hintgroup>
</stringresponse>
</problem>

```

12 Publishing Your Resources

In order to make the content you've created available for use in courses, you must publish your content. LON-CAPA provides an easy interface for publishing your content pages, problem resources, and sequences. You can specify title, author information, keywords, and other metadata. LON-CAPA uses this metadata for many things, and it's important to fill the metadata out as accurately as possible.

12.1 What is Metadata?

Metadata is *data about data*. Metadata can often be thought of as a label on some bit of information that can be useful to people or computer programs trying to use the data. Without metadata, the person or computer trying to use the original information would have to guess what the original data is about.

When resources are published at least title, subject and keywords should be provided so that the resource could be found easily.

For example, if you create a problem and neglect to say in the title or subject of the problem what it is about, then a human who wants to use that problem would have to read the problem itself to see what it was about. This is much more difficult than just reading a title. A computer trying to do the same thing would be out of luck; it is too stupid to understand the problem statement at all.

Another example of metadata is the `<title>` tag of a web page, which usually shows up in the title bar of the browser. That is information about the web page itself and is not actually part of the web page. People use the title information when they bookmark a page. Search engines use it as a clue about the content of the web page.

12.2 Publishing a Resource

To publish a resource, log in and choose your Author role. You should see something like the resource listing of the figure 4. Use the **Actions** dropdown to select **Publish** for the resource you wish to publish. You will get a metadata screen that should look something like the "Publishing Metadata Screen" figure. Fill out the form. If you are creating resources

Title:

Author(s):

Subject:

Keywords:

<input type="checkbox"/> anterophase	<input type="checkbox"/> cellular	<input type="checkbox"/> class	<input type="checkbox"/> concept	<input type="checkbox"/> discussion	<input type="checkbox"/> division	<input type="checkbox"/> fourb	<input type="checkbox"/> hint	<input type="checkbox"/> mitosis	<input type="checkbox"/> oneb	<input type="checkbox"/> onec	<input checked="" type="checkbox"/> phase
<input type="checkbox"/> phases	<input type="checkbox"/> questions	<input type="checkbox"/> recall	<input type="checkbox"/> statement	<input type="checkbox"/> text	<input type="checkbox"/> threea	<input type="checkbox"/> twoa	<input type="checkbox"/> twob	<input type="checkbox"/> wednesday			

Additional Keywords:

Notes:

Abstract:

LANGUAGE:

Publisher/Owner:

COPYRIGHT/DISTRIBUTION:

Figure 22: Publishing Metadata Screen

that may be used in several courses, you should talk with the other authors and establish some sort of standard title and subject scheme in advance.

Language is the language the problem is written in. **Publisher/Owner** is the LON-CAPA user who owns the problem.

Keywords and **Abstract** are more information about the problem.

The **Keywords** are words that are strongly connected to your problem; for instance a physics problem about a pulley might include “pulley” as a key word. LON-CAPA pulls out words used in the text of the resource for you so you can just click on their check boxes to make them keywords. **Additional keywords** allows you to add any keyword to your problem that are not actually in the problem. For instance, on that same problem a physicist might add the keyword “statics”, even though it doesn’t appear in the original problem, because Physics uses that as a classification of problem type. **Additional Keywords** are also useful when publishing graphics.

You need to set the copyright and distribution permissions in the **COPYRIGHT/DISTRIBUTION** drop-down. This setting controls who is allowed to use your resource as follows:

- **System Wide - can be used for any courses system wide** is the default. The content can be used for any course within the network, regardless of the domain. Instructors all over the world can find your content and use it in their courses. Once an instructor has selected a resource, the students in the course can have access to it.
- **Domain - Limited to courses in the domain published** means that only courses running in the same domain as you can use your content.
- **Private - visible to author only** is not supported anymore. Use *Customized right of use* instead.
- **Public - no authentication required** means anyone can find and use the resource - even without being logged in to the system.
- **Customized right of use** means that access to the resource is controlled by a separate Custom Rights file. This file needs to be specified during publication. You can edit a Custom Rights file in your author space, and need to publish it like any other file. Any number of your resource can point at the same Custom Rights file - if you want to change access rights for all of them, you just need to change and re-publish this one file.

Not all of these choices may be visible, depending on the nature of the resource.



Now when you click **Finalize Publication**, your resource will be published and usable (unless you set the distribution to “private”).

If you’re following this as a tutorial, publish your resources so we can use them in the next section.

13 Printing Your Resources

13.1 Printing from Authoring Space

To print a resource, do the following:


1. The  icon will only be accessible when you are looking at one of your resources for your course.
2. Click  to access the Print Helper, which will help you create a PDF document.

The Print helper will guide you through the process of preparing a PDF document of the resource. If you see error message when trying to prepare a PDF file, then you will need to contact the author of the problem which contains that printing error.

Printing involves a translation of your XML file into LaTeX and from there to PDF. Some of the XML tags have a set of special print options, see 13.3. Sometimes translations require special considerations, see 13.4.

13.2 Printing a Subdirectory of Problems

Authors or co-authors will typically choose to organize their authoring spaces into sub-directories. To generate a PDF containing all problems in an entire directory or sub-directory, follow the steps below:

1. First, select one of the problems to view in the sub-directory to be printed.
2. Click  **Print** in the Functions menu if using the Inline menu, or the PRT button on the Remote Control.
3. Select the option to print ‘Selected Problems’ from current subdirectory.’
4. Optionally, choose to print with Answers. You can choose to print with either the default two column output or with one column.
5. Click the Next button.
6. On the next screen, select which problems you want to print by either clicking on one of the Select buttons or individually checking which problems you want to select.
7. Click the Next button.
8. As long as there were no errors in any of the problems, you should now see a link to a PDF file that you can download and view.

If an error occurs with just one of the problems then printing of all selected problems in the sub-directory will fail. LON-CAPA will make a guess at which problem(s) had errors. You will need to troubleshoot and fix those problems before the entire directory can be printed. Information about common print errors is available at 13.4.

13.3 Tips for Improving Print Output

Here you can find some useful tips how to make your printing output looking prettier.

Print output oriented attributes of standard HTML/LON CAPA tags

- 13.3.1 <h1>-<hN> **TeXsize** attribute
- 13.3.1 <basefont> **TeXsize** attribute
- 13.3.1 **TeXsize** attribute
- 13.3.2 <hr> **TeXwidth** attribute
- 13.3.2 <table> **TeXwidth** attribute
- 13.3.3 <table> **TeXDropEmptyColumns** attribute
- 13.3.2 <td> **TeXwidth** attribute
- 13.3.2 <th> **TeXwidth** attribute
- 13.3.4 **TeXwidth** attribute
- 13.3.4 **TeXheight** attribute
- 13.3.4 **TeXwrap** attribute This attribute controls how the generated LaTeX attempts to wrap text around figures when a horizontal alignment has been requested in the IMG tag. Unfortunately, L^AT_EX is not really built to do this and there are no known perfect solutions. This attribute has two possible values:
 - texwrap - (the default) uses the texwrap environment to attempt to get text to wrap around the picture. This requires either a “left” or “right” alignment, and works well in most cases.
 - parpic - uses the picins package \parpic to attempt to get text to wrap around the image. This method places the remainder of the text of the paragraph containing the picture to the left or right of the picture. This scheme has two drawbacks: If the remainder of the paragraph text is insufficient to fill the area to the side of the image, the text from the following paragraph will run through the image, parpic also seems to not do a good job of honoring the end of the page, and images can spill below the page footers generated by Lon-CAPA.

13.3.1 TeXsize Attribute

TeXsize attribute in <h1>-<hN>, <basefont>, and tags declares the size of LaTeX fonts used in printing.

Possible values of **TeXsize** attribute:

tiny	smallest
scriptsize	very small
footnotesize	smaller
small	small
normalsize	normal
large	large
Large	larger
LARGE	even larger
huge	still larger
Huge	largest

Note, that all parameters coincide with standard LaTeX commands for changing font size though you do not escape them.

Examples:

```
<basefont size="4" TeXsize="Large" />
<font color="#FFFFFF" TeXsize="small">
<h1 align="center" TeXsize="Huge">
```

13.3.2 TeXwidth Attribute

TeXwidth attribute allows you to specify the width of

- the table in `<table>` tag
- the table cell in `<td>` or `<th>` tags
- the length of the line in `<hr>` tag

You can use the following units:

mm	
cm	=10 mm
in	=25.4 mm
pt	=0.35 mm
pc	=4.22 mm

Examples:

```
<hr TeXwidth="2 cm">
<td TeXwidth="1 in">
```

13.3.3 TeXDropEmptyColumns Attribute

TeXDropEmptyColumns attribute allows you to suppress printing of empty columns in table. This option is useful when you have deal with big tables (very often nested) with a lot of empty columns. Situation is typical in chemistry where tables are used for visualization of chemical reactions.

Example:

```
<table TeXDropEmptyColumns="yes">
```

13.3.4 Image TeX Attributes

- **Image Url** contains the URL of the image to be inserted in the problem. You may enter a URL or click “Select” to choose an image that has already been uploaded to your authoring space, or click “Search”
- **Description** contains a textual description of the image. If the image cannot be rendered by the target browser, this description is displayed instead.
- **width (pixel)** allows you to set the width of the image, in pixels, as it will be displayed in a web browser.
- **height (pixel)** allows you to set the height of the image, in pixels, as it will be displayed in a web browser.
- **TeXwidth (mm)** Allows you to set the width of the image, in mm, as it will be rendered into the \LaTeX document used to print the problem.
- **TeXheight (mm)** Allows you to set the height of the image, in mm, as it will be rendered into the \LaTeX document used to print the problem.
- **TeXwrap** Allows you to select how the \LaTeX document will attempt to wrap text around a horizontally aligned image (See Alignment below).

parbox \newline and \parbox will be used to place the image. This method ensures that text will not be wrapped on top of the image, however very little text will appear next to the image itself.

parpic The `picins` package **\parpic** command will be used to place the image. This will wrap the remainder of the paragraph containing the picture around the image. If, however, there is insufficient text to fill the space to the left or right of the image, the next paragraph may be wrapped on top of the image. In addition, **\parpic** does not always honor the end of the page, causing the image to extend below the page footer.

- **Alignment** Specifies the alignment of the image relative to the enclosing text paragraph:

bottom The image will be aligned so that its bottom will be at the baseline of the surrounding text.

middle The image will be aligned so that its center-line will be at the baseline of the surrounding text.

top The image will be aligned so that its top will be at the baseline of the surrounding text.

left The image will be placed so that it is at the left of the surrounding text. The surrounding text will fill in the region to the right of the image.

right The image will be placed so that it is at the right of the surrounding text. The surrounding text will fill in the region to the left of the image.

13.3.5 TeX Type Attribute

TeXtype attribute is responsible for the definition of the type of LaTeX list environment used during printing of available options. Possible values of this attribute:

TeXtype attribute is responsible for the definition of the type of LaTeX list environment used during printing of available options. Possible values of this attribute:

value	example of list
1	[1.] First Item [2.] Second Item [3.] Third Item
A	[A.] First Item [B.] Second Item [C.] Third Item
a	[a.] First Item [b.] Second Item [c.] Third Item
i	[i.] First Item [ii.] Second Item [iii.] Third Item

Examples:

```
<radiobuttonresponse TeXtype="1">
```

```
<radiobuttonresponse TeXtype="A">
```

13.3.6 TeX Itemgroup Width Attribute

TeXitemgroupwidth attribute allows you to specify the width of table with items for matching. The value of this attribute defines the width in percents with respect to text line width.

```
<matchresponse TeXitemgroupwidth="40%">
```

13.3.7 TeX Layout Attribute

TeXlayout attribute governs the way how available options are displayed when printed - either vertically (attribute value - "vertical") or horizontally (attribute value - "horizontal").

Examples:

```
<optionresponse TeXlayout="horizontal">
<optionresponse TeXlayout="vertical">
```

13.4 Troubleshooting PDF Errors

When you print a LON-CAPA resource, the XML of your resource is translated into LaTeX. The LaTeX is then processed and turned into a PDF document which can be displayed with your browser's Acrobat plugin and subsequently printed.

There are several problems that crop up both due to limitations in the XML to LaTeX translation and due to differences in the model used by web browsers to render HTML and LaTeX to compose print pages. This document provides information about some of these problems and, where possible, solutions, and tricks to work around them. If you have a printing trick or a problem and would like to report it, please go to <http://bugs.lon-capa.org> and register a bug report.

General information about printing within LonCAPA is also available in section 13.1. Some types of problems that may occur include:

The print rendition of some Perl functions looks ugly

In particular these functions are:

- &prettyprint
- &dollarformat
- &xmllparse
- &chemparse

To make these two functions work correctly within the print translator, it is necessary to wrap them within a `<display>` tag. For example:

```
<p>
If I had <display>&prettyprint(100,'$2f')</display>
</p>
```

Note that the `<display>` tags must be tightly wrapped around the function call or you will get a syntax error in web presentation mode. For additional information about cases where you must use `<display>`, see "Variables with tags don't print correctly" below.

Image placement and alignment and text wrapping is wrong

Unfortunately this is due to a large difference between the LaTeX and HTML page layout model. In HTML images are placed exactly where you ask them to be placed. In LaTeX,

images are considered *floats*, which LaTeX will place for you. Some of the common html tricks, using tables e.g. to control text wrapping around figures, will not always work in print mode; especially if the text is to the right side of the figure in the table.

The alignment choice affects whether or not the print rendering engine attempts to get text to wrap around the image. With `align="right"` or `align="left"`, the print rendering engine attempts to use the *wrapfigure* environment to place text around the figure at the appropriate side. If a figure is in a table, then the print engine, by default, the print engine will use *wrapfigure*, set the alignment to “right” unless you override it. Otherwise, the default alignment is “bottom” as it is for html, and no wrapping will occur.

`\parpic` style wrapping is also available by specifying `TeXwrap="parpic"` in the `img` tag. In some limited cases this gives a better result.

Other print specific `` tag attributes are available (see section 13.3).

Variables with tags don't print correctly

If a variable contains XML, in general it is necessary to force the XML parser to make a pass over the contents of the contents of the variable prior to rendering the section of the resource that contains that substitution. When output, those variables must be bracketed inside of `<display>` `</display>` tags. For example:

```
<problem>
<script type="loncapa/perl">
$a = &xmlparse('<br />');
</script>
<startouttext />
<p>This is a break <display>$a</display> and then some more text</p>
<endouttext />
</problem>
```

Without the `xmlparse` call and the `display` tag bracketing the variable, this problem will display on the web just fine, but print incorrectly.

14 Authoring Maps: Sequences and Pages

In order to create a useful course, we need to arrange our raw materials so that students can use them.

14.1 Authoring Sequences

A **sequence** is a series of resources that can be navigated using the **NAV** remote control button, or by using the arrow keys on the remote control.

To create a Sequence resource, create a new resource as described in the 'Authoring Content in LON-CAPA' section (5). This is a “sequence” resource so the URL must end in “.sequence”. After you enter in the URL ending in “.sequence”, you should see a screen as in figure 23. You can use either the advanced editor or the simplified editor.

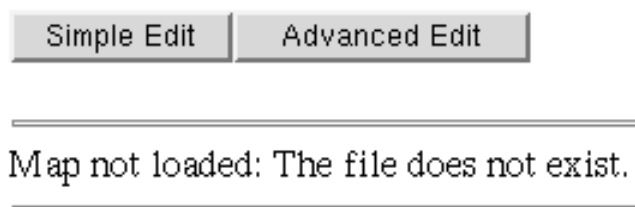


Figure 23: Map Editor Selection

Map not loaded: The file does not exist.

/~jerf/totallyNew.sequence

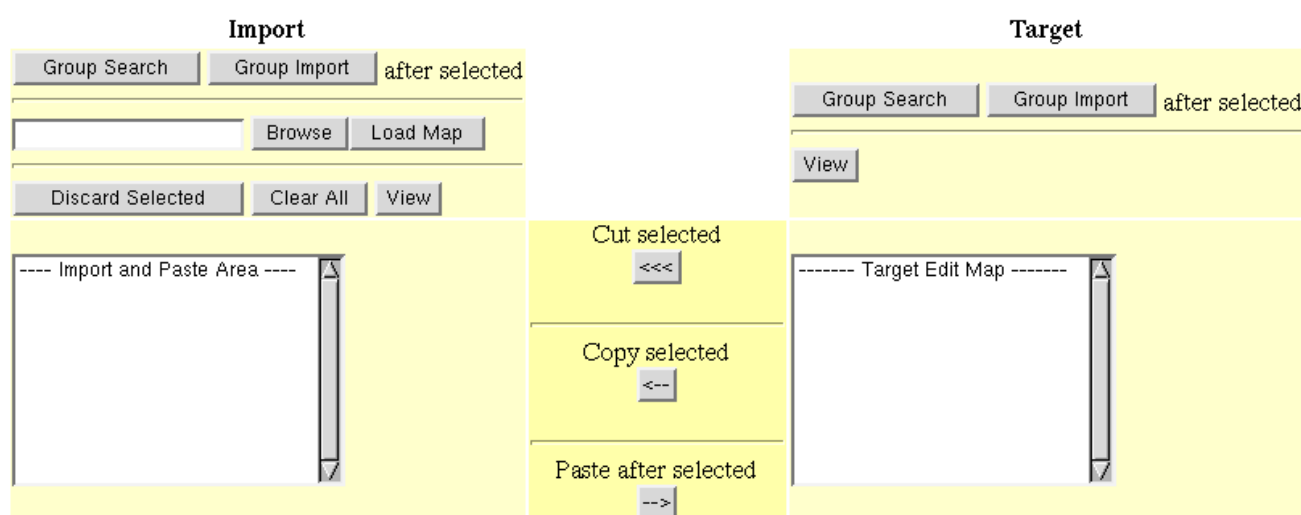


Figure 24: Simple Map Editor

14.2 Authoring a Simple .sequence With The Simple Editor

After creating a new .sequence resource and getting the editor selection prompt (as in the “Simple Map Editor” figure), click the **Simple Edit** button to get to the simple map editor, which appears in the figure.

The Simple Editor can create .sequences and .pages which are linear, which means they have no branches or conditions.

On the right side of the simple editor is the **Target**, which represents the map you are currently building. On the left side is the **Import** area, which represents a work area you can use for your convenience to load and manipulate resources you may wish to include in your map. Using the three buttons in the middle of the screen, you can cut things out of the Target (top button), copy from the Target to the Import (middle button), and copy from the Import to the Target (bottom button).

You can do a Group Search and a Group Import on both sides of the screen. A Group



Figure 25: Initial Map Editor

Search allows you to run a search, then import selected results from that search either directly into your Map or into your Import space. Checkboxes will appear next to the results in the Group Search, and you can click the resources you wish to add to your map in the order that you want them added. After you select the resources, you will be presented with a screen that allows you to change their order. You will then be able to import the selected resources and work with them.

A Group Import works in a similar fashion, but allows you to use the LON-CAPA network browser to select your resources.

On the Import side, you can also browse for another Map, and load the resources used in that map into your Import workspace. You can also discard the selected resources, clear all the resources, and view the selected resources by using the buttons on the Import side of the screen.

Both list boxes support standard multi-select mechanisms as used in your OS.

14.3 Authoring a Simple .sequence With The Advanced Editor

After creating a new .sequence resource and getting the editor selection prompt (23), click the **Advanced Edit** button to get to the advanced map editor. You should see the initial map editor as shown in the “Initial Map Editor” figure. Note there are two windows: One is the workspace and one is a secondary window which will contain information as you add resources.

Click the **Start** box. You’ll see what is shown in the “After clicking **Start** in the Map Constructor” figure. Click **Link Resource** in the secondary window then click on the **Finish** box. After that, click **Straighten**. You should see something looking like the “Straightened Map” figure. This creates a simple map that flows from beginning to end.

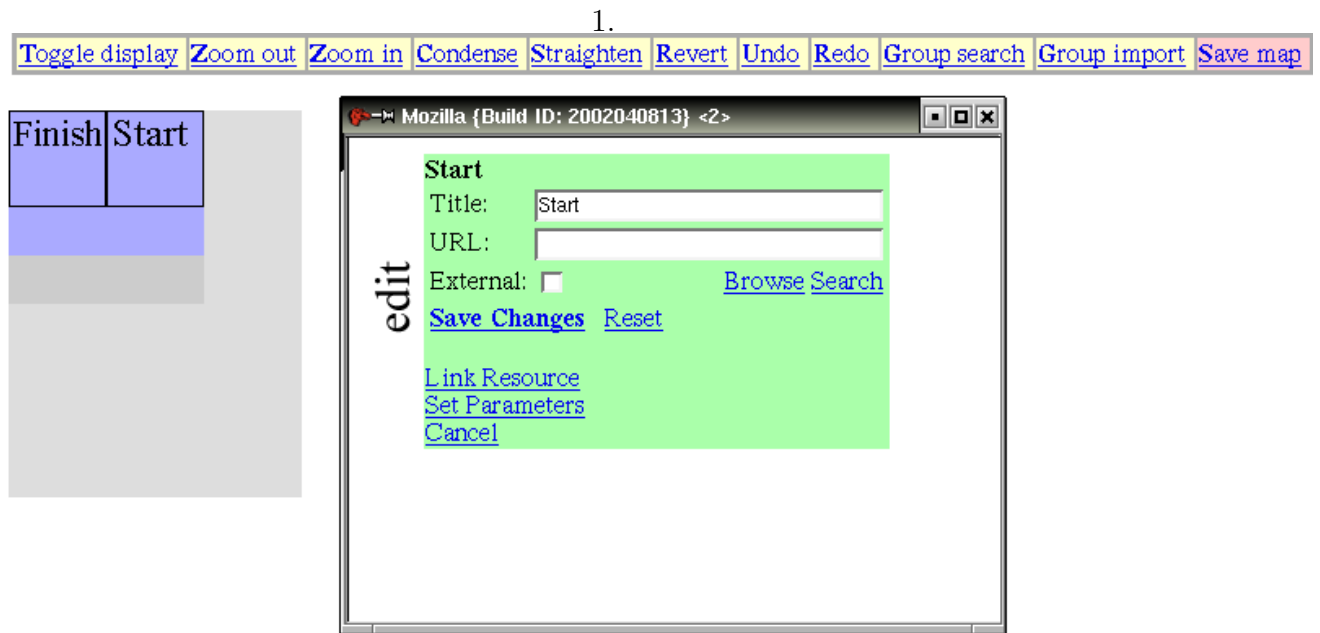
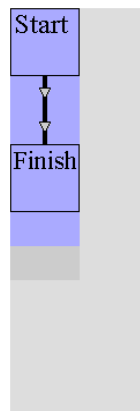
Figure 26: After clicking **Start** in the Map Constructor

Figure 27: Straightened Map

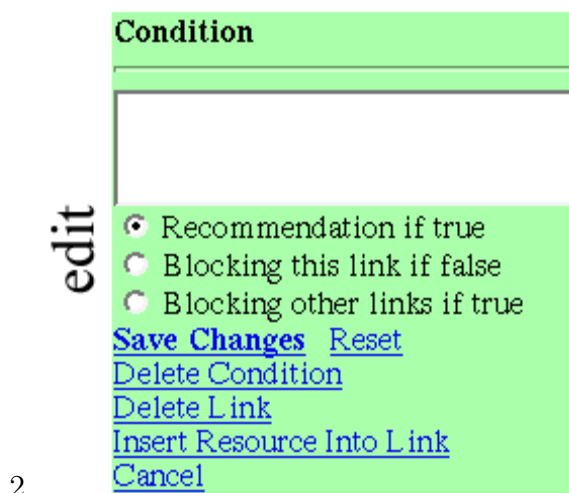


Figure 28: Inserting a Resource

The LearningOnline With CAPA Network Directory Browser

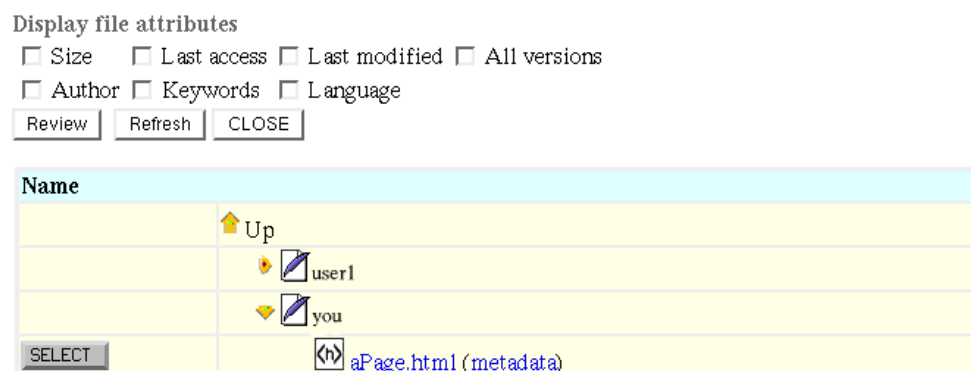


Figure 29: Network Directory Browser

To insert a resource into the flow, click the black line with two arrows, seen between the **Start** and **Finish** boxes in the “Straightened Map” figure. In the secondary window, you will see something like the “Inserting a Resource” figure. Click **Insert Resource Into Link**. A new resource box will appear in the link. Click the resource, which will have the label **Res**.

3. Click **Browse** and the **Network Directory Browser** will appear, as shown in the “Network Directory Browser” figure. Press the **SELECT** button that is next to the resource you want to place in the chosen resource box. Once you’ve done that, if you look back at the window that popped up when you clicked on **New Resource**, you’ll see something like the “Resource Chosen” figure. You can type the **URL** and **Title** into the secondary window if you prefer, following the format you see when you’ve successfully browsed to a resource. After you click **Save Changes**, your changes will be set and the icons for the resource will appear in the **Res** box, as shown in figure

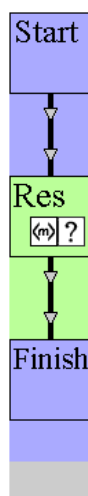


Figure 30: Resource Chosen

30. Click **Save Map** in the bar above your map to save the map.

Clicking on the left icon for a resource will open a new browser window with an informational page about that resource. Clicking on the right icon for a resource will open a new browser window and take you to the rendering of that resource.

4. Repeat steps two and three for as many resources as you'd like to bind together into one page. You can insert the new resources anywhere you'd like.
5. When you are done adding resources, click the **Save Map** link to save the map.

In addition to manually adding in resources, the Advanced Editor also has the ability to import resources in the same way that the Simple Editor can: From a LON-CAPA network browser window, from a Group Search, or from another Map.

The Advanced Editor has many more capabilities which you can explore.

14.4 Page Maps

Creating a Page map is the same as creating a Sequence map, except that when choosing the name of the resource, the URL will end with “.page”. This way, all resources you add in the map editor will appear on one page together. Pages are often used to connect problems in a homework set.

14.5 Courses: Top-level Sequence

In order to view sequences, they need to be part of a **course**.

Courses have a Top-level map which defines the whole course. This Top-Level map will often contain maps corresponding to homework assignments, chapters, or units. To view your maps, you will need to make them part of a course.

Create a new Course

Course Title

Top-level Map

 [Browse](#)

Course ID/Number (optional)

Course Coordinator

Username:

Domain: ▾

Figure 31: Creating a New Course

When a course is first created a top-level map will be generated. This top-level map will be an “internal” sequence file which is managed using the **Course Editor**. It is also possible to set a course’s Top-Level map to be a published sequence, but that is very rarely done. In that case the Course Editor would no longer be used to manage course content.

In the typical case a course’s “Main Content” area corresponds to the top-level map, and the Course Editor is used to add, remove or re-order content within it. Users assigned the role of Course Coordinator in a course will upload content or import published content into the course. Folders and sub-folders can be created using the Course Editor, and content uploaded/imported into them.

The content can include published maps (both sequences and pages). Sequences will appear as folders in the Course Contents listing. See the Course Coordination manual for more information.

Your Domain Coordinator may have assigned you rights to request courses. If so, visit your roles screen, and select the “Request course” item in the Functions menu. If not, you should contact your Domain Coordinator to ask about course creation.

15 Tags Used in XML Authoring

It is assumed that the reader is already familiar with the basic terminology of XML. If not, it is recommended that you read http://www.w3schools.com/xml/xml_syntax.asp to acquire a basic understanding of how to read and write XML.

LON-CAPA uses a very simple subset of XML and there is a lot you do *not* need to know, including but not limited to: CDATA, DTDs, namespaces, and stylesheets. If you search for XML resources on the Internet yourself, you do not need to read about those things to learn

how LON-CAPA uses XML for problems.

15.1 Response Tags

Response tags are the tags used by LON-CAPA to indicate what a student should enter into the system, such as a string answer, clicking on a picture, typing in a formula, etc. They are the core tags of homework problems; a homework problem without at least one response tag is not really a homework problem.

Simple examples of the more complicated tags are available as templates for you to choose from when creating a new problem in your Authoring Space.

15.1.1 numericalresponse

stringresponse implements a string answer. An internal **textline** tag (see 15.5) is necessary for the student's response to go in. It can check the string for either case or order. Possible attributes are:

- **answer**: required. Specifies the correct answer, either a perl list or scalar.
- **type**: optional. Specifies how the string is checked (like the CAPA styles). Possible values are:
 - **cs**: case sensitive, order important.
 - **ci**: case insensitive, order important.
 - **mc**: case insensitive, order unimportant. The mnemonic for this option is “multiple choice”, which is how it was used in CAPA: To allow the user to specify choices from a multiple choices problem, as in “adce”, meaning parts a, d, c, and e are true. Order didn't matter in such a problem. In LON-CAPA, using **option-response** with True and False foils would be preferable, but this will remain supported for easier CAPA to LON-CAPA conversion.

15.1.2 imageresponse

imageresponse implements a image-click answer. **imageresponse** tags should contain a **foilgroup** tag, which contain **foil** tags. Each **foil** tag can contain:

- **image**: required. The delimited text should correspond to a published image resource. Example: `<image>/res/adm/includes/templates/man1.jpg</image>`. The following image formats are recommended - gif, jpg or png. Other formats may work, but there may be printing or display issues. The image should only appear once per foil.
- **rectangle**: required. The delimited text specifies a rectangular area that is correct, specified as `(x1,y1)-(x2,y2)`, where x1, x2, y1, and y2 are number corresponding to the x and y coordinates of two corners that define a rectangle which specifies where the right answer for this foil is located on the image. For example, `(0,0)-(100,200)` will specify that a rectangle 100 pixels wide and 200 pixels tall, situated in the upper

left of the image, is correct. At least one rectangle is required; multiple rectangles may be specified.

- **text:** required. The delimited text is printed before the image is shown on the screen. This text is typically used to describe to the student what they are expected to click on.

15.1.3 optionresponse

optionresponse implements a “select from these choices” style question. The choices are specified by the instructor and use the foil structure tags as described in 15.3, with this additional addition:

- **foilgroup:** required to have an *options* attribute which should be a perl list of possible options for the student.

15.1.4 radiobuttonresponse

radiobuttonresponse implements a true/false question with one correct answer. It uses the foil structure tags as described in 15.3, but the *value* of a foil can only be **true**, **false**, or **unused**.

15.1.5 dataresponse

dataresponse is an advanced type of response tag that implements a simple data storage and needs an input tag, such as **textline**, to work correctly. Possible attributes are:

- **name:** internal name for the value. It will have the part id and response id added to it.
- **type:** type of data stored in this response field. It should be one of the types supported by `parameter.html`
- **display:** string that will be used to describe the field when interfacing with humans.

15.1.6 externalresponse

externalresponse is an advanced type of response tag that implements the ability to have an external program grade a response. It expects either a **textline** or **textfield** inside the tag. Possible attributes are:

- **url:** url to submit the answer form to. It does not need to be a LON-CAPA machine.
- **answer:** data to post in the form element `LONCAPA_correct_answer` to the remote site.
- **form:** hash variable name that will be submitted to the remote site as a HTTP form.

The form sent will consist of

- **LONCAPA_student_response** full text of what the student entered in the entry field
- **LONCAPA_correct_answer** contents of the answer attribute
- **LONCAPA_language** specified language encoding of the requesting resource
- **all items in the form attribute** if any of these clash with the above, the above values will overwrite the value in the form attribute

The response of the remote server needs to be in XML as follows:

- **loncapagrade**: takes no attributes, but must surround the response.
- **awardsdetail**: required. The delimited text inside must be one of the detailed results that appears in the data storage documentation. CVS:loncapa/doc/homework/datastorage, look for **resource.partid.responseid.awardsdetail**.
- **message**: optional message to have shown to the student.

Example:

```
<loncapagrade>
  <awardsdetail>INCORRECT</awardsdetail>
  <message>
A message to be shown to the students
  </message>
</loncapagrade>
```

15.1.7 Attributes For All Response Tags

These response tag attributes are used by all response tags:

- **id**: If this isn't set, it will be set during the publication step. It is used to assign parameter names in a way that can be tracked if an instructor modifies by hand.
- **name**: optional. If set, it will be used by the resource assembly tool when one is modifying parameters.

15.2 responseparam and parameter

If **responseparam** appears, it should be inside of a response tag. It defines an externally adjustable parameter for the question, which the question can then use to allow other users to customize the problem for their courses without changing the source code of the problem. Possible attributes are:

- **default**: required. Specifies a default value for the parameter.
- **name**: required. Specifies an internal name for the parameter.

- **type**: required. Specifies the type of parameter: **tolerance**, **int**, **float**, **string**, or **date**.
- **description**: string describing the parameter. This is what is used to talk about a parameter outside of a problem.

parameter is exactly the same as **responseparam**, but should appear outside of a response tag.

15.3 Foil Structure Tags

All tags that implement a foil structure have an optional arg of *max* that controls the maximum number of total foils to show.

- **foilgroup**: required. Must surround all foil definitions.
- **foil**: required. The foil is defined by what is delimited by the **foil** tag.
- **conceptgroup**: optional. Surrounds a collection of **foil**. When a problem is displayed, only one of the contained **foil** is selected for display. It has one required attribute **concept**.

15.4 Hint Tags

All of these tags must appear inside a response tag:

- **hintgroup**: Tag that surrounds all of a hint.
- **hintpart**: Tag to implement conditional hints. It has a required argument **on**. When a hint tag named the same as the **on** attribute evaluates to be correct, the **hintpart** will show. If no other **hintpart** is to show then all hintparts with an **on** value set to “default” will show.
- **numericalhint**: It has all the arguments that **numericalresponse** does, and the required attribute **name** which should be set to the value of which **hintpart** will be shown.
- **stringhint**: It has all the arguments that **stringresponse** does, and the required attribute **name** which should be set to the value of which hintpart will be shown.
- **formulahint**: It has all the arguments that **formularesponse** does, and the required attribute **name** which should be set to the value of which hintpart will be shown.
- **optionhint**: The required attribute **name** should be set to the value of which **hintpart** will be shown.

- **radiobuttonhint**: The required attribute **name** should be set to the value of which **hintpart** will be shown, and the attribute **answer** should be at least a two element list: first the type (**foil** or **concept**) and then either the foil name(s) or the concept string(s), e.g., “(‘foil’, ‘greaterthan’, ‘equal’)” if the condition should be triggered by the foils named “greaterthan” or “equal.”
- **customhint**: The required attribute **name** should be set to the value of which **hintpart** will be shown. Define the hint condition within an **answer** block inside of the **customhint** block. The condition is defined like how an answer is defined in **customresponse** where you need to return EXACT_ANS to indicate when customhint criteria are met.

15.5 Input Tags

This group of tags implements a mechanism for getting data for students. They will usually be used by a response tag.

- **textfield**: Creates a large text input box. If data appears between the start and end tags, the data will appear in the textfield if the student has not yet made a submission. Additionally, it takes two attributes: **rows** and **cols**, which control the height and width of the text area respectively. It defaults to 10 rows and 80 columns.
- **textline**: Creates a single line input element. It accepts one attribute **size** which controls the width of the textline, defaulting to 20.

Both the **textfield** and **textline** input tags support a **spellcheck** attribute. If present, the text the user types is spellchecked when focus leaves the input field. The value of this attribute specifies the language in which the spellcheck is performed. e.g. `<textfield spellcheck='en' />` creates a text field that is spellchecked in English while `<textline spellcheck='de' />` creates a text field that is spellchecked in German.

15.6 Output Tags

This group of tags generates useful output.

- **algebra**: Typesets algebraic expressions

```
<algebra>2x^y+sqrt(3/x^2)</algebra>
```

Expressions are displayed using the math expression display mechanism defined in the user's preferences. The default is tth. See the section below concerning the `<m>` tag for more information on that as well as on the attribute **display**.

- **chem**: Typesets chemical equation

```
<chem>O2 + 2H2 -> 2H2O</chem>
```

- **num**: Typesets a number formatted in scientific notation, fixed point, fixed point with commas, fixed point with commas and dollar sign, or in significant digits.

```
<num format="2E">31454678</num> results in 3.15 x 107
<num format="2f">31454678</num> results in 31454678.00
<num format="4g">31454678</num> results in 3.145 x 107
<num format="4g">314.54678</num> results in 314.5
<num format=",2f">31454678</num> results in 31,454,678.00
<num format="$2f">31454678</num> results in $31,454,678.00
<num format="2s">31454678</num> results in 31000000
<num format=",2s">31454678</num> results in 31,000,000
```

- **parse**: to display the parsed view of a variable's contents

```
<script type="loncapa/perl">
  $table='<table>';
  for ($i=1;$i<=10;$i++) {
    $table.='<tr><td>'. $i. '</td><td>'.&random(1,10,1). '</td></tr>';
  }
  $table.='</table>';
</script>
<parse>$table</parse>
```

- **standalone**: Everything in between the start and end tag is shown only on the web and only if the resource is not part of a course.
- **displayduedate**: This will insert the current due date if one is set in the document. It is generated to be inside a table of 1x1 elements. The displayduedate tag accepts The following attributes:

style="plain" Makes the due date appear without any boxing. If the parameter value is other than "*plain*", or if the **style** parameter is omitted, the due date will be displayed within a box.

format="fmt_string" Allows you to control the format of the due date. "*fmt_string*" is an arbitrary string that can contain any of the following formatting items:

%a Replaced by the abbreviated weekday name according to the current locale.

%A Replaced by the full weekday name according to the current locale.

%b The abbreviated month name according to the current locale.

%B The full month name according to the current locale.

%c The preferred date and time representation for the current locale (the default format string is just this).

%C The century number as a two digit integer

%d The day of the month as a decimal number. Leading zeroes are shown for single digit day numbers.

%D Equivalent to %m/%d/%y
 %e Like %d but a leading zero is replaced by a space.
 %F Equivalent to %Y-%m-%d
 %G The four digit year number.
 %g The two digit year number.
 %H The hour as a two digit number in the range 00 through 23.
 %I The hour as a two digit number in the range 00 through 12.
 %j The day of the year in the range 001 through 366.
 %k The hour (24 hour clock), single digits are preceded by a blank.
 %l Like %k but using a 12 hour clock.
 %m The month as a two digit decimal number in the range 01 through 12.
 %M The minute as a two digit decimal number in the range 00 through 59.
 %n A newline character.
 %p AM or PM depending on the time value.
 %P am or pm.
 %r The time in am or pm notation.
 %R Time in 24 hour notation (%H:%M). See also %T below.
 %s Number of seconds since midnight of January 1, 1970.
 %S The second as a decimal number in the range 00 through 59.
 %t A horizontal tab character.
 %T The time in 24 hour notation (%H:%M:%S).
 %u Day of the week as a decimal number with Monday as 1.
 %U The week number of the current year in the range 00 through 53. Week 1 is the week containing the first Sunday of the year.
 %V Same as %U but week 1 is the first week with at least 4 days, with Monday being the first day of a week.
 %w Day of the week as a decimal integer in the range 0 through 7, Sunday is 0.
 %W Week number of the current year in the range 00 through 53, where the first Monday of the year is the first day of week 01.
 %x The preferred date notation in the current locale without the time.
 %X The preferred time notation in the current locale without the date.
 %y The year as a decimal number without the century (range 00 through 99).
 %Y The year as a decimal number including the century.
 %% A % character.
 %+ Date and time in the form returned by the Unix date command.

- **displaytitle**: This will insert the title of the problem from the metadata of the problem. Only the first **displaytitle** in a problem will show the title; this allows clean usage of **displaytitle** in LON-CAPA style files.

- **window**: This creates a link that when clicked shows the intervening information in a pop-up window. By default the window will be 500 pixels wide and 200 pixels tall, and the link text will be a superscript * (so as to look like a footnote). These can be changed using the attributes
 - **width** controls the starting width of the popup window
 - **height** controls the starting height of the popup window
 - **linktext** the text that should appear as the link that causes the creation of the window
 - **printtext** the text that should appear instead of a footnote when printed

When printing, the included text will get turned into a real footnote.

- **windowlink**: This creates a link to a resource that comes up in a pop-up window. The link will be the intervening information between the start and the end tag. By default the window will be 500 pixels wide and 200 pixels tall.
 - **width** controls the starting width of the popup window
 - **height** controls the starting height of the popup window
 - **href** the address
- **togglebox**: This creates a toggling box that can be clicked open and close.
 - **heading** heading text of the box, by default no heading
 - **headerbg** background color of the header, by default white
 - **showtext** the text that appears to make the box visible, by default the translation of 'show'
 - **hidetext** the text that appears to hide the box again, by default the translation of 'hide'

When printing, the included text will be rendered in a visible box.

- **m**: The inside text is \LaTeX , and is converted to HTML (or MathML) on the fly. The default is to convert to the display mechanism that the user has selected in preferences. This can be overridden by setting the attribute **display** to one of “**tth**” or “**jsMath**” or “**mimetex**” which will force a specific display mechanism. Note, however, that setting the attribute **display** to **jsmath** is generally discouraged as it works best if users have installed jsmath fonts on their computer. See <http://www.math.union.edu/~dpvc/jsmath/>.

If you want variables inside of this tag to be evaluated before the tex gets converted, then use `eval=“on”` . For example, `<m eval=“on”>$ $eqn $</m>`, will evaluate the variable `$eqn` first and then run it through the TTH converter. Anytime you use a variable inside of the `m` tag, you will want to set `eval` to `on`.

For example, put the following in a script in the resource:

```
$eqn = "$a+$b";
```

```
$eqn = s/\+\/-\/g;
```

and in a text area, you can type:

```
<m eval='on'>$ $eqn $</m>
```

You will get the equation rendered with no $+-$, no matter what value b may take on.

- **randomlabel**: This shows a specified image with images or text labels randomly assigned to a set of specific locations. Those locations may also have values assigned to them. A hash is generated that contains the mapping of labels to locations, labels to values, and locations to values. Example:

```
<randomlabel bgimg="URL" width="12" height="45" texwidth="50">
  <labelgroup name="GroupOne" type="image">
    <location x="123" y="456" value="10" />
    <location x="321" y="654" value="20" />
    <location x="213" y="546" value="13" />
    <label description="TEXT-1">IMG-URL</label>
    <label description="TEXT-2">IMG-URL</label>
    <label description="TEXT-3">IMG-URL</label>
  </labelgroup>
  <labelgroup name="GroupTwo" type="text">
    <location x="12" y="45" />
    <location x="32" y="65" />
    <location x="21" y="54" />
    <label>TEXT-1</label>
    <label>TEXT-2</label>
    <label>TEXT-3</label>
  </labelgroup>
</randomlabel>
```

Possible attributes are:

- **bgimg**: Either a fully qualified URL for an external image or a LON-CAPA resource. It supports relative references (`../images/apicture.gif`). The image must either be a GIF or JPEG.
- **width**: The width of the image in pixels.
- **height**: The height of the image in pixels.
- **texwidth**: The width of the image in millimeters.

- **problemtype**: This tag allows you to show or hide output based on what the problem-type parameter is set to in the course. For example:

```
<problemtype mode="show" for="exam,survey">
<startouttext />
The formula for the circumference of a circle is 2*pi*r
<endouttext />
</problemtype>
```

Will only show the output text when the problem is set to the type of exam or survey in the course. The attribute for mode can be set to show or hide. The attribute for for can be problem, exam, survey, or practice.

15.7 Internal Tags

- **labelgroup**: One is required, but multiple are allowed. This declares a group of locations and labels associated with them. Possible attributes are:
 - **name**: This is the name of the group. A hash with this name will be generated holding the mappings for later use in the problem. For each location a value will be set for which label is there (EX. `$hash{'1'}="TEXT-2"`). For locations with values, the hash will contain 2 items, a location to value mapping (`$hash{'value.1'}=10`), and a label to value mapping (`$hash{'labelvalue.2'}=10`). For all image style of labels there will also be a label description to label URL mapping (`$hash{'image.2'}=IMG-URL`). The entry **numlocations** will also be set to the total number of locations that exist (Note: locations and labels start counting from one.)
 - **type**: the type of labels in this group, either **'image'** or **'text'**
 - **location**: declares a location on the image that a label should appear at. Possible attributes are:
 - * **x**: The x value of the location in pixels.
 - * **y**: The y value of the location in pixels.
 - * **value**: An optional scalar value to associate at this location.
 - * **label**: Declaration of a label. If this is a **text** type label, the internal text should be the text of the label (HTML is not currently supported); if this is an **image** type of label, the internal text must be a LON-CAPA resource specification, and the description filed must be set. Possible attributes are:
 - **description**: Required field for image labels. It will be used when setting values in the hash.

15.8 Scripting Tags

- **display**: The intervening Perl script is evaluated in the safe space and the return value of the script replaces the entire tag.

- **import**: This causes the parse to read in the file named in the body of the tag and parse it as if the entire text of the file had existed at the location of the tag.
- **parserlib**: The enclosed filename contains definitions for new tags.
- **script**: If the attribute **type** is set to “loncapa/perl” the enclosed data is a Perl script which is evaluated inside the Perl safe space. The return value of the script is ignored.
- **scriptlib**: The enclosed filename contains Perl code to run in the safe space.
- **block**: This has a required argument **condition** that is evaluated. If the condition is true, everything inside the tag is evaluated; otherwise, everything inside the block tag is skipped.
- **notsolved**: Everything inside the tag is skipped if the problem is “solved”.
- **postanswerdate**: Everything inside the tag is skipped if the problem is before the answer date.
- **preduedate**: Everything inside the tag is skipped if the problem is after the due date.
- **randomlist**: The enclosed tags are parsed in a stable random order. The optional attribute **show=“N”** restricts the number of tags inside that are actually parsed to no more than N. N can equal the total tags inside. The randomlist tag can be used to randomize problem parts by wrapping the <part> tags with a randomlist tag. Note that when randomlist wraps <part> tags, that all students will work all parts only if **show=“N”** where N is the total number of parts wrapped. When N is less than the total number of parts wrapped, there will be gaps in the assessment chart, and also in the table of submissions for each student, corresponding to those parts which are never available to that particular student. For more examples see Authoring Library Files section 10.1.
- **solved**: Everything inside the tag is skipped if the problem is “not solved”.
- **while**: This implements a while loop. The required attribute **condition** is a Perl scriptlet that when evaluated results in a true or false value. If true, the entirety of the text between the whiles is parsed. The condition is tested again, etc. If false, it goes to the next tag.

15.9 Structure Tags

These tags give the problem a structure and take care of the recording of data and giving the student messages.

- **problem**: This must be the first tag in the file. This tag sets up the header of the webpage and generates the submit buttons. It also handles due dates properly.
- **part**: This must be below **problem** if it is going to be used. It does many of the same tasks as **problem**, but allows multiple separate problems to exist in a single file.

- **startouttext** and **endouttext**: These tags are somewhat special. They must have no internal text and occur in pairs. Their use is to mark up the problem so the web editor knows what sections should be edited in a plain text block on the web.
- **comment**: This tag allows one to comment out sections of code in a balanced manner, or to provide a comment description of how a problem works. It only shows up for the edit target, stripped out for all other targets.

16 <script> Tag

16.1 Supported Script Functions

This is a list of functions that have been written that are available in the Safe space scripting environment inside a problem:

- $\sin(x)$, $\cos(x)$, $\tan(x)$
- $\text{asin}(x)$, $\text{acos}(x)$, $\text{atan}(x)$, $\text{atan2}(y,x)$
- $\log(x)$, $\log_{10}(x)$
- $\exp()$, $\text{pow}(x,y)$, $\text{sqrt}(x)$
- $\text{abs}(x)$, $\text{sgn}(x)$
- $\text{erf}(x)$, $\text{erfc}(x)$
- $\text{ceil}(x)$, $\text{floor}(x)$
- $\text{min}(\dots)$, $\text{max}(\dots)$
- $\text{factorial}(n)$
- $N\%M$ (modulo function)
- $\sinh(x)$, $\cosh(x)$, $\tanh(x)$
- $\text{asinh}(x)$, $\text{acosh}(x)$, $\text{atanh}(x)$
- $\text{roundto}(x,n)$
- $\text{cas}(s,e,l)$
- $\text{web}(\text{"a"}, \text{"b"}, \text{"c"})$ or $\text{web}(a,b,c)$
- $\text{html}(\text{"a"})$ or $\text{html}(a)$
- $j_0(x)$, $j_1(x)$, $j_n(n,x)$, $j_v(y,x)$
- $y_0(x)$, $y_1(x)$, $y_n(n,x)$, $y_v(y,x)$

- random
- choose
- tex(“a”,”b”) or tex(a,b)
- var_in_tex(a)
- to_string(x), to_string(x,y)
- class(), sec(), classid()
- name(), firstname(), middlename(), lastname(), student_number()
- check_status(partid)
- open_date(partid), due_date(partid), answer_date(partid)
- open_date_epoch(partid), due_date_epoch(partid), answer_date_epoch(partid)
- submission(partid,responseid,version)
- currentpart()
- sub_string()
- array_moments(array)
- format(x,y),prettyprint(x,y,target),dollarformat(x,target)
- languages
- map(...)
- capareponse_check
- capareponse_check_list
- parameter_setting(name,partid)
- EXT()
- stored_data(name,partid)
- wrong_bubbles(correct,lower,upper,step,@given)

We also support these functions from Math::Cephes

bdtr: Binomial distribution
 bdtrc: Complemented binomial distribution
 bdtri: Inverse binomial distribution
 btdtr: Beta distribution
 chdtr: Chi-square distribution
 chdtrc: Complemented Chi-square distribution
 chdtri: Inverse of complemented Chi-square distribution
 fdtr: F distribution
 fdtrc: Complemented F distribution
 fdtri: Inverse of complemented F distribution
 gdtr: Gamma distribution function
 gdtrc: Complemented gamma distribution function
 nbdtr: Negative binomial distribution
 nbdtrc: Complemented negative binomial distribution
 nbdtri: Functional inverse of negative binomial distribution
 ndtr: Normal distribution function
 ndtri: Inverse of Normal distribution function
 pdtr: Poisson distribution
 pdtrc: Complemented poisson distribution
 pdtri: Inverse Poisson distribution
 stdtr: Student's t distribution
 stdtri: Functional inverse of Student's t distribution

Please see Math::Cephes for more information

16.2 Script Variables

- \$external::target - set to the current target the xml parser is parsing for
- \$external::part - set to the *id* of the current problem <part>; zero if there are no <part>
- \$external::gradestatus - set to the value of the current resource.partid.solved value
- \$external::datestatus - set to the current status of the clock either CLOSED, CAN_ANSWER, CANNOT_ANSWER, SHOW_ANSWER, or UNCHECKEDOUT
- \$external::randomseed - set to the number that was used to seed the random number generator
- \$pi - set to PI
- \$rad2deg - converts radians to degrees
- \$deg2rad - converts degrees to radians

16.3 Table: LON-CAPA Functions

LON-CAPA Function	Description
&sin(\$x), &cos(\$x), &tan(\$x)	Trigonometric functions where x is in radians. \$x can be a pure number, i.e., you can call &sin(3.1415)
&asin(\$x), &acos(\$x), &atan(\$x), &atan2(\$y,\$x)	Inverse trigonometric functions. Return value is in radians. For asin and acos the value of x must be between -1 and 1. The atan2 returns a value between -pi and pi the sign of which is determined by y. \$x and \$y can be pure numbers
&log(\$x), &log10(\$x)	Natural and base-10 logarithm. \$x can be a pure number
&exp(\$x), &pow(\$x,\$y), &sqrt(\$x)	Exponential, power and square root, i.e., ex, xy and /x. \$x and \$y can be pure numbers
&abs(\$x), &sgn(\$x)	Abs takes the absolute value of x while sgn(x) returns 1, 0 or -1 depending on the value of x. For x>0, sgn(x) = 1, for x=0, sgn(x) = 0 and for x<0, sgn(x) = -1. \$x can be a pure number
&erf(\$x), &erfc(\$x)	Error function. erf = 2/sqrt(pi) integral (0,x) et-sq and $erf(x) = 1.0 - erf(x)$. \$x can be a pure number
&ceil(\$x), &floor(\$x)	Ceil function returns an integer rounded up whereas floor function returns an integer rounded down. If x is an integer then it returns the value of the integer. \$x can be a pure number
&min(...), &max(...)	Returns the minimum/ maximum value of a list of arguments if the arguments are numbers. If the arguments are strings then it returns a string sorted according to the ASCII codes
&factorial(\$n)	Argument (n) must be an integer else it will round down. The largest value for n is 170. \$n can be a pure number
\$N%M	N and M are integers and returns the remainder (in integer) of N/M. \$N and \$M can be pure numbers
&sinh(\$x), &cosh(\$x), &tanh(\$x)	Hyperbolic functions. \$x can be a pure number
&asinh(\$x), &acosh(\$x), &atanh(\$x)	Inverse hyperbolic functions. \$x can be a pure number
&format(\$x,'nn')	Display or format \$x as nn where nn is nF or nE or nS and n is an integer.

LON-CAPA Function	Description
&prettyprint(\$x,'nn','optional target')	Note that that tag <num> can be used to do the same thing. Display or format \$x as nn where nn is nF or nE or nS and n is an integer. Also supports the first character being a \$, it then will format the result with a a call to &dollarformat() described below. If the first character is a , it will format it with commas grouping the thousands. In S mode it will fromat the number to the specified number of significant figures and display it in F mode. In E mode it will attempt to generate a pretty $x10^3$ rather than a E3 following the number, the 'optional target' argument is optional but can be used to force &prettyprint to generate either 'tex' output, or 'web' output, most people do not need to specify this argument and can leave it blank.
&dollarformat(\$x,'optional target')	Reformats \$x to have a \$ (or \ \$ if in tex mode) and to have , grouping thousands. The 'optional target' argument is optional but can be used to force &prettyprint to generate either 'tex' output, or 'web' output, most people do not need to specify this argument and can leave it blank.
Option 1 - \$best = &languages() Option 2 - @all = &languages() Option 3 - \$best = &languages(\@desired_languages) Option 4 - @all = &languages(\@desired_languages)	Returns the best language to use, in the first two options returns the languages codes in the preference order of the user. In the second two examples returns the best matches from a list of desired language possibilities.
&roundto(\$x,\$n)	Rounds a real number to n decimal points. \$x and \$n can be pure numbers
&cas(\$s,\$e,\$l)	Evaluates the expression \$e inside the symbolic algebra system \$s. Currently, the Maxima symbolic math system ('maxima') and the R statistical computing system ('R') are implemented. \$l is an optional comma-separated list of libraries. Example: &cas('maxima','diff(sin(x)/cos(x),x,2)')

LON-CAPA Function	Description
<code>&implicit_multiplication(\$f)</code>	Adds mathematical multiplication operators to the formula expression <code>\$f</code> where only implicit multiplication is used. Example: <code>&implicit_multiplication('2(b+3c)')</code> returns $2*(b+3*c)$
<code>&web("a","b","c")</code> or <code>&web(\$a,\$b,\$c)</code>	Returns either a, b or c depending on the output medium. a is for plain ASCII, b for tex output and c for html output
<code>&html("a")</code> or <code>&html(\$a)</code>	Output only if the output mode chosen is in html format
<code>&j0(\$x)</code> , <code>&j1(\$x)</code> , <code>&jn(\$m,\$x)</code> , <code>&jv(\$y,\$x)</code>	Bessel functions of the first kind with orders 0, 1 and m respectively. For <code>jn(m,x)</code> , m must be an integer whereas for <code>jv(y,x)</code> , y is real. <code>\$x</code> can be a pure number. <code>\$m</code> must be an integer and can be a pure integer number. <code>\$y</code> can be a pure real number
<code>&y0(\$x)</code> , <code>&y1(\$x)</code> , <code>&yn(\$m,\$x)</code> , <code>&yv(\$y,\$x)</code>	Bessel functions of the second kind with orders 0, 1 and m respectively. For <code>yn(m,x)</code> , m must be an integer whereas for <code>yv(y,x)</code> , y is real. <code>\$x</code> can be a pure number. <code>\$m</code> must be an integer and can be a pure integer number. <code>\$y</code> can be a pure real number
<code>&random(\$l,\$u,\$d)</code>	Returns a uniformly distributed random number between the lower bound, l and upper bound, u in steps of d. d is optional. If omitted, a step of 1 is used. <code>\$l</code> , <code>\$u</code> and <code>\$d</code> can be pure numbers.
<code>&choose(\$i,...)</code>	Choose the i th item from the argument list. i must be an integer greater than 0 and the value of i should not exceed the number of items. <code>\$i</code> can be a pure integer

LON-CAPA Function	Description
<p>Option 1 - <code>&map(\$seed,[\ \$w,\ \$x,\ \$y,\ \$z],[\$a,\$b,\$c,\$d])</code> or Option 2 - <code>&map(\$seed,\ @mappedArray,[\$a,\$b,\$c,\$d])</code> Option 3 - <code>@mappedArray =</code> <code>&map(\$seed,[\$a,\$b,\$c,\$d])</code> Option 4 - <code>(\$w,\$x,\$y,\$z) =</code> <code>&map(\$seed,\ @a)</code> Option 5 - <code>@Z = &map(\$seed,\ @a)</code> where <code>\$a='A'</code> <code>\$b='B'</code> <code>\$c='B'</code> <code>\$d='B'</code> <code>\$w, \$x, \$y, and \$z</code> are variables</p>	<p>Assigns to the variables <code>\$w</code>, <code>\$x</code>, <code>\$y</code> and <code>\$z</code> the values of the <code>\$a</code>, <code>\$b</code>, <code>\$c</code> and <code>\$c</code> (A, B, C and D). The precise value for <code>\$w</code> .. depends on the seed. (Option 1 of calling map). In option 2, the values of <code>\$a</code>, <code>\$b</code> .. are mapped into the array, <code>@mappedArray</code>. The two options illustrate the different grouping. Options 3 and 4 give a consistent way (with other functions) of mapping the items. For each option, the group can be passed as an array, for example, <code>[\$a,\$b,\$c,\$d] => \ @a</code>. And Option 5 is the same as option 4, where the array of results is saved into a single array rather than an array of scalar variables.</p>
<p>Option 1 - <code>&rmap(\$seed,[\ \$w,\ \$x,\ \$y,\ \$z],[\$a,\$b,\$c,\$d])</code> or Option 2 - <code>&rmap(\$seed,\ @rmappedArray,[\$a,\$b,\$c,\$d])</code> Option 3 - <code>@rmapped_array =</code> <code>&rmap(\$seed,[\$a,\$b,\$c,\$d])</code> Option 4 - <code>(\$w,\$x,\$y,\$z) =</code> <code>&rmap(\$seed,\ @a)</code> Option 5 - <code>@Z = &map(\$seed,\ @a)</code> where <code>\$a='A'</code> <code>\$b='B'</code> <code>\$c='B'</code> <code>\$d='B'</code> <code>\$w, \$x, \$y, and \$z</code> are variables</p>	<p>The rmap functions does the reverse action of map if the same seed is used in calling map and rmap.</p>
<p><code>\$a=&xmlparse(\$string)</code></p>	<p>You probably should use the tag <code><parse></code> instead of this function. Runs the internal parser over the argument parsing for display. Warning This will result in different strings in different targets. Don't use the results of this function as an answer.</p>
<p><code>&tex(\$a,\$b), &tex("a","b")</code></p>	<p>Returns a if the output mode is in tex otherwise returns b</p>

LON-CAPA Function	Description
&var_in_tex(\$a)	Equivalent to tex("a","")
&to_string(\$x), &to_string(\$x,\$y)	If x is an integer, returns a string. If x is real than the output is a string with format given by y. For example, if x = 12.3456, &to_string(x,".3F") = 12.345 and &to_string(x,".3E") = 1.234E+01.
&class(), &sec(), &classid()	Returns null string, class descriptive name, section number, class id, set number and null string.
&name(), &student_number(), &firstname(), &middlename(), &lastname()	Return the full name in the following format: lastname, firstname initial. Student_number returns the student 9-alphanumeric string. The functions firstname, middlename, and lastname return just that part of the name. If undefined, the functions return null.
&check_status(\$partid)	Returns a number identifying the current status of a part. True values mean that a part is "done" (either unanswerable because of tries exhaustion, or correct) or a false value if a part can still be attempted. If \$part is unspecified, it will check either the current <part>'s status or if outside of a <part>, check the status of previous <part>. The full set of return codes are: 'undef' means it is unattempted, 0 means it is attempted and wrong but still has tries, 1 means it is marked correct, 2 means they have exceeded maximum number of tries, 3 means it is after the answer date.
&open_date(\$partid), &due_date(\$partid), &answer_date(\$partid)	Problem open date, due date and answer date in local human-readable format. Part 0 is chosen if \$partid is omitted.
&open_date_epoch(\$partid), &due_date_epoch(\$partid), &answer_date_epoch(\$partid)	Problem open date, due date and answer date in seconds after the epoch (UTC), which can be used in calculations.

LON-CAPA Function	Description
&submission(\$partid,\$responseid,\$version,\$encode,\$cleanupnum)	Returns what the student submitted for response \$responseid in part \$partid. You can get these IDs from the XML-code of the problem. Use 0 as \$partid for problems without parts. \$version is optional and returns the \$version-th submission of the student that was graded. If \$version is 0 or omitted, the latest submission is returned. \$encode is also optional and allows the author to explicitly encode the returned string. It's up to the author to take care of properly escaping all characters which might be interpreted by the browser. \$cleanupnum is also optional, and supports clean-up of the retrieved submission. It is a reference to a hash, with one or more of the following: exponent => 1, comma => 1, letterforzero => 1, spaces => 1, format => 'ns' (where n is an integer, i.e., number of significant digits). For example, to convert a student submission of 11,300 to 11300 include { comma => 1, } as the fifth arg.
¶meter_setting(\$name,\$partid)	Returns the parameter setting \$name. Partid is optional.
&stored_data(\$name,\$partid)	Returns the stored data \$name. Partid is optional.
&wrong_bubbles(\$correct,\$lower,\$upper,\$step,@gr)	Returns an array that can be used for wrong answers in numerical responses. The first argument is the correct answer, the next arguments are the lower and upper boundaries for the bubbles, as well as the step size. The next argument is an optional array of wrong answers that should be included.
¤tpart()	Returns the ID of the current part.
Not implemented	Get and set the random seed.
&sub_string(\$a,\$b,\$c) perl substr function. However, note the differences	Retrieve a portion of string a starting from b and length c. For example, \$a = "Welcome to LON-CAPA"; \$result=&sub_string(\$a,4,4); then \$result is "come"
@arrayname Array is intrinsic in perl. To access a specific element use \$arrayname[\$n] where \$n is the \$n+1 element since the array count starts from 0	"xx" can be a variable or a calculation.

LON-CAPA Function	Description
@B=&array_moments(@A)	Evaluates the moments of an array A and place the result in array B[i] where i = 0 to 4. The contents of B are as follows: B[0] = number of elements, B[1] = mean, B[2] = variance, B[3] = skewness and B[4] = kurtosis.
&min(@Name), &max(@Name)	In LON-CAPA to find the maximum value of an array, use &max(@arrayname) and to find the minimum value of an array, use &min(@arrayname)
undef @name	To destroy the contents of an array, use
@return_array=&random_normal (\$item_cnt,\$seed,\$av,\$std_dev)	Generate \$item_cnt deviates of normal distribution of average \$av and standard deviation \$std_dev. The distribution is generated from seed \$seed
@return_array=&random_beta (\$item_cnt,\$seed,\$aa,\$bb) NOTE: Both \$aa and \$bb MUST be greater than 1.0E-37.	Generate \$item_cnt deviates of beta distribution. The density of beta is: $X^{($aa-1)} * (1-X)^{($bb-1)} / B($aa,$bb)$ for $0 < X < 1$.
@return_array=&random_gamma (\$item_cnt,\$seed,\$a,\$r) NOTE: Both \$a and \$r MUST be positive.	Generate \$item_cnt deviates of gamma distribution. The density of gamma is: $(a^{**}r) / \text{gamma}(r) * X^{**}($r-1) * \exp(-$a*X)$.
@return_array=&random_exponential (\$item_cnt,\$seed,\$av) NOTE: \$av MUST be non-negative.	Generate \$item_cnt deviates of exponential distribution.
@return_array=&random_poisson (\$item_cnt,\$seed,\$mu) NOTE: \$mu MUST be non-negative.	Generate \$item_cnt deviates of poisson distribution.
@return_array=&random_chi (\$item_cnt,\$seed,\$df) NOTE: \$df MUST be positive.	Generate \$item_cnt deviates of chi_square distribution with \$df degrees of freedom.
@return_array=&random_noncentral_chi (\$item_cnt,\$seed,\$df,\$nonc) NOTE: \$df MUST be at least 1 and \$nonc MUST be non-negative.	Generate \$item_cnt deviates of noncentral_chi_square distribution with \$df degrees of freedom and noncentrality parameter \$nonc.
@return_array=&random_f (\$item_cnt,\$seed,\$dfn,\$dfd) NOTE: Both \$dfn and \$dfd MUST be positive.	Generate \$item_cnt deviates of F (variance ratio) distribution with degrees of freedom \$dfn (numerator) and \$dfd (denominator).
@return_array=&random_noncentral_f (\$item_cnt,\$seed,\$dfn,\$dfd,\$nonc) NOTE: \$dfn must be at least 1, \$dfd MUST be positive, and \$nonc must be non-negative.	Generate \$item_cnt deviates of noncentral F (variance ratio) distribution with degrees of freedom \$dfn (numerator) and \$dfd (denominator). \$nonc is the noncentrality parameter.

LON-CAPA Function	Description
@return_array=&random_multivariate_normal (\$item_cnt,\$seed,\@mean,\@covar) NOTE: \@mean should be of length p array of real numbers. \@covar should be a length p array of references to length p arrays of real numbers (i.e. a p by p matrix.	Generate \$item_cnt deviates of multivariate_normal distribution with mean vector \@mean and variance-covariance matrix.
@return_array=&random_multinomial (\$item_cnt,\$seed,@p) NOTE: \$item_cnt is rounded with int() and the result must be non-negative. The number of elements in @p must be at least 2.	Returns single observation from multinomial distribution with \$item_cnt events classified into as many categories as the length of @p. The probability of an event being classified into category i is given by ith element of @p. The observation is an array with length equal to @p, so when called in a scalar context it returns the length of @p. The sum of the elements of the observation is equal to \$item_cnt.
@return_array=&random_permutation (\$seed,@array)	Returns @array randomly permuted.
@return_array=&random_uniform (\$item_cnt,\$seed,\$low,\$high) NOTE: \$low must be less than or equal to \$high.	Generate \$item_cnt deviates from a uniform distribution.
@return_array=&random_uniform_integer (\$item_cnt,\$seed,\$low,\$high) NOTE: \$low and \$high are both passed through int(). \$low must be less than or equal to \$high.	Generate \$item_cnt deviates from a uniform distribution in integers.
@return_array=&random_binomial (\$item_cnt,\$seed,\$nt,\$p) NOTE: \$nt is rounded using int() and the result must be non-negative. \$p must be between 0 and 1 inclusive.	Generate \$item_cnt deviates from the binomial distribution with \$nt trials and the probability of an event in each trial is \$p.
@return_array=&random_negative_binomial (\$item_cnt,\$seed,\$ne,\$p) NOTE: \$ne is rounded using int() and the result must be positive. \$p must be between 0 and 1 exclusive.	Generate an array of \$item_cnt outcomes generated from negative binomial distribution with \$ne events and the probability of an event in each trial is \$p.

The &EXT() external function is extremely powerful, and is used to access parameters and submission values. It can be used within scripts and also within cell formulas in the grading spreadsheet. Some examples can be found by browsing in the repository to /res/msu/albertel/test/ext_examples.html. The &EXT() function can be used to obtain values for the same parameters as are retrieved by some of the other (newer) helper functions summarized in the table above, such as &firstname() which is equivalent to &EXT('environment.firstname'), and ¶meter_setting(\$name,\$partid) is equivalent to &EXT('resource.'.\$partid.'.\$name). In such cases the newer (specialized) functions are preferred to &EXT() on the basis of ease of use.

16.4 Table: CAPA vs. LON-CAPA Function Differences

CAPA Functions	LON-CAPA	Differences (if any)
$\sin(x)$, $\cos(x)$, $\tan(x)$	<code>&sin(\$x)</code> , <code>&cos(\$x)</code> , <code>&tan(\$x)</code>	
$\operatorname{asin}(x)$, $\operatorname{acos}(x)$, $\operatorname{atan}(x)$, $\operatorname{atan2}(y,x)$	<code>&asin(\$x)</code> , <code>&acos(\$x)</code> , <code>&atan(\$x)</code> , <code>&atan2(\$y,\$x)</code>	
$\log(x)$, $\log_{10}(x)$	<code>&log(\$x)</code> , <code>&log10(\$x)</code>	
$\exp(x)$, $\operatorname{pow}(x,y)$, \sqrt{x}	<code>&exp(\$x)</code> , <code>&pow(\$x,\$y)</code> , <code>&sqrt(\$x)</code>	
$\operatorname{abs}(x)$, $\operatorname{sgn}(x)$	<code>&abs(\$x)</code> , <code>&sgn(\$x)</code>	
$\operatorname{erf}(x)$, $\operatorname{erfc}(x)$	<code>&erf(\$x)</code> , <code>&erfc(\$x)</code>	
$\operatorname{ceil}(x)$, $\operatorname{floor}(x)$	<code>&ceil(\$x)</code> , <code>&floor(\$x)</code>	
$\min(\dots)$, $\max(\dots)$	<code>&min(\dots)</code> , <code>&max(\dots)</code>	
$\operatorname{factorial}(n)$	<code>&factorial(\$n)</code>	
$N\%M$	<code>\$N\%\$M</code>	
$\sinh(x)$, $\cosh(x)$, $\tanh(x)$	<code>&sinh(\$x)</code> , <code>&cosh(\$x)</code> , <code>&tanh(\$x)</code>	
$\operatorname{asinh}(x)$, $\operatorname{acosh}(x)$, $\operatorname{atanh}(x)$	<code>&asinh(\$x)</code> , <code>&acosh(\$x)</code> , <code>&atanh(\$x)</code>	
<code>/DIS(\$x,"nn")</code>	<code>&format(\$x,'nn')</code>	The difference is obvious.
Not in CAPA	<code>&prettyprint(\$x,'nn','optional target')</code>	
Not in CAPA	<code>&dollarformat(\$x,'optional target')</code>	
Not in CAPA	<code>&languages(@desired_languages)</code>	
$\operatorname{roundto}(x,n)$	<code>&roundto(\$x,\$n)</code>	
Not in CAPA	<code>&cas(\$s,\$e)</code>	
Not in CAPA	<code>&implicit_multiplication(\$f)</code>	
<code>web("a","b","c")</code> or <code>web(a,b,c)</code>	<code>&web("a","b","c")</code> or <code>&web(\$a,\$b,\$c)</code>	
<code>html("a")</code> or <code>html(a)</code>	<code>&html("a")</code> or <code>&html(\$a)</code>	
$\operatorname{jn}(m,x)$	<code>&j0(\$x)</code> , <code>&j1(\$x)</code> , <code>&jn(\$m,\$x)</code> , <code>&jv(\$y,\$x)</code>	In CAPA, <code>j0</code> , <code>j1</code> and <code>jn</code> are contained in one function, <code>jn(m,x)</code> where <code>m</code> takes the value of 0, 1 or 2. <code>jv(y,x)</code> is new to LON-CAPA.
$\operatorname{yn}(m,x)$	<code>&y0(\$x)</code> , <code>&y1(\$x)</code> , <code>&yn(\$m,\$x)</code> , <code>&yv(\$y,\$x)</code>	In CAPA, <code>y0</code> , <code>y1</code> and <code>yn</code> are contained in one function, <code>yn(m,x)</code> where <code>m</code> takes the value of 0, 1 or 2. <code>yv(y,x)</code> is new to LON-CAPA.
$\operatorname{random}(l,u,d)$	<code>&random(\$l,\$u,\$d)</code>	In CAPA, all the 3 arguments must be of the same type. However, now you can mix the type
$\operatorname{choose}(i,\dots)$	<code>&choose(\$i,\dots)</code>	

CAPA Functions	LON-CAPA	Differences (if any)
/MAP(seed;w,x,y,z;a,b,c,d)	<p>Option 1 - $\&\text{map}(\\$seed, [\backslash \\$w, \backslash \\$x, \backslash \\$y, \backslash \\$z], [\\$a, \\$b, \\$c, \\$d])$ or Option 2 - $\&\text{map}(\\$seed, \backslash @mappedArray, [\\$a, \\$b, \\$c, \\$d])$ Option 3 - @mappedArray = $\&\text{map}(\\$seed, [\\$a, \\$b, \\$c, \\$d])$ Option 4 - ($\\$w, \\$x, \\$y, \\z) = $\&\text{map}(\\$seed, \backslash @a)$ where $\\$a='A'$ $\\$b='B'$ $\\$c='B'$ $\\$d='B'$ $\\$w, \\$x, \\$y$, and $\\$z$ are variables</p>	In CAPA, the arguments are divided into three groups separated by a semicolon ;. In LON-CAPA, the separation is done by using [] brackets or using an array @a. Note the backslash (\) before the arguments in the second and third groups.
rmap(seed;a,b,c,d;w,x,y,z)	<p>Option 1 - $\&\text{rmap}(\\$seed, [\backslash \\$w, \backslash \\$x, \backslash \\$y, \backslash \\$z], [\\$a, \\$b, \\$c, \\$d])$ or Option 2 - $\&\text{rmap}(\\$seed, \backslash @rmappedArray, [\\$a, \\$b, \\$c, \\$d])$ Option 3 - @rmapped_array = $\&\text{rmap}(\\$seed, [\\$a, \\$b, \\$c, \\$d])$ Option 4 - ($\\$w, \\$x, \\$y, \\z) = $\&\text{rmap}(\\$seed, \backslash @a)$ where $\\$a='A'$ $\\$b='B'$ $\\$c='B'$ $\\$d='B'$ $\\$w, \\$x, \\$y$, and $\\$z$ are variables</p>	In CAPA, the arguments are divided into three groups separated by a semicolon ;. In LON-CAPA, the separation is done by using [] brackets (with create an unnamed vector reference) or using an array @a. Note the backslash (\) before the arguments in the second and third groups (Which cause Perl to send to variable locations rather than the variable values, similar to a C pointer).
NOT IMPLEMENTED IN CAPA	$\$a=\&\text{xmlparse}(\$string)$	New to LON-CAPA
tex(a,b), tex("a","b")	$\&\text{tex}(\$a, \$b), \&\text{tex}("a", "b")$	
var_in_tex(a)	$\&\text{var_in_tex}(\$a)$	
to_string(x), to_string(x,y)	$\&\text{to_string}(\$x), \&\text{to_string}(\$x, \$y)$	
capa_id(), class(), section(), set(), problem()	$\&\text{class}(), \&\text{sec}()$	capa_id(), set() and problem() are no longer used. Currently, they return a null value.

CAPA Functions	LON-CAPA	Differences (if any)
name(), student_number()	&name(), &student_number()	
open_date(), due_date(), answer_date()	&open_date(), &due_date(), &answer_date()	Output format for time is changed slightly. If pass noon, it displays ..pm else it displays ..am. So 23:59 is displayed as 11:59 pm.
get_seed(), set_seed()	Not implemented	
sub_string(a,b,c)	&sub_string(\$a,\$b,\$c) perl substr function. However, note the differences	Perl intrinsic function, substr(string,b,c) starts counting from 0 (as opposed to 1). In the example to the left, substr(\$a,4,4) returns "ome".
array[xx]	@arrayname Array is intrinsic in perl. To access a specific element use \$arrayname[\$n] where \$n is the \$n+1 element since the array count starts from 0	In LON-CAPA, an array is defined by @arrayname. It is not necessary to specify the dimension of the array.
array_moments(B,A)	@B=&array_moments(@A)	In CAPA, the moments are passed as an array in the first argument whereas in LON-CAPA, the array containing the moments are set equal to the function.
array_max(Name), array_min(Name)	&min(@Name), &max(@Name)	Combined with the min and max functions defined earlier.
init_array(Name)	undef @name	Use perl intrinsic undef function.
random_normal (return_array,item_cnt,seed,av,std_dev)	@return_array=&random_normal (\$item_cnt,\$seed,\$av,\$std_dev)	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_beta (return_array,item_cnt,seed,aa,bb)	@return_array=&random_beta (\$item_cnt,\$seed,\$aa,\$bb) NOTE: Both \$aa and \$bb MUST be greater than 1.0E-37.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_gamma (return_array,item_cnt,seed,a,r)	@return_array=&random_gamma (\$item_cnt,\$seed,\$a,\$r) NOTE: Both \$a and \$r MUST be positive.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_exponential (return_array,item_cnt,seed,av)	@return_array=&random_exponential (\$item_cnt,\$seed,\$av) NOTE: \$av MUST be non-negative.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.

CAPA Functions	LON-CAPA	Differences (if any)
random_poisson (return_array,item_cnt,seed,mu)	@return_array=&random_poisson (\$item_cnt,\$seed,\$mu) NOTE: \$mu MUST be non-negative.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_chi (return_array,item_cnt,seed,df)	@return_array=&random_chi (\$item_cnt,\$seed,\$df) NOTE: \$df MUST be positive.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_noncentral_chi (return_array,item_cnt,seed,df,nonc)	@return_array=&random_noncentral_chi (\$item_cnt,\$seed,\$df,\$nonc) NOTE: \$df MUST be at least 1 and \$nonc MUST be non-negative.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
NOT IMPLEMENTED IN CAPA	@return_array=&random_f (\$item_cnt,\$seed,\$dfn,\$dfd) NOTE: Both \$dfn and \$dfd MUST be positive.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_noncentral_f (\$item_cnt,\$seed,\$dfn,\$dfd,\$nonc) NOTE: \$dfn must be at least 1, \$dfd MUST be positive, and \$nonc must be non-negative.	New to LON-CAPA
NOT DOCUMENTED IN CAPA	@return_array=&random_multivariate (\$item_cnt,\$seed,\@mean,\@covar) NOTE: \@mean should be of length p array of real numbers. \@covar should be a length p array of references to length p arrays of real numbers (i.e. a p by p matrix.	Not in CAPA. Note the backslash before the \@mean and \@covar arrays.
NOT IMPLEMENTED IN CAPA	@return_array=&random_multinomial (\$item_cnt,\$seed,@p) NOTE: \$item_cnt is rounded with int() and the result must be non-negative. The number of elements in @p must be at least 2.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_permutation (\$seed,@array)	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_uniform (\$item_cnt,\$seed,\$low,\$high) NOTE: \$low must be less than or equal to \$high.	New to LON-CAPA

CAPA Functions	LON-CAPA	Differences (if any)
NOT IMPLEMENTED IN CAPA	@return_array=&random_uniform_integers(\$item_cnt,\$seed,\$low,\$high) NOTE: \$low and \$high are both passed through int(). \$low must be less than or equal to \$high.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_binomial(\$item_cnt,\$seed,\$nt,\$p) NOTE: \$nt is rounded using int() and the result must be non-negative. \$p must be between 0 and 1 inclusive.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_negative_binomial(\$item_cnt,\$seed,\$ne,\$p) NOTE: \$ne is rounded using int() and the result must be positive. \$p must be between 0 and 1 exclusive.	New to LON-CAPA

17 Bridge Task

Bridge Tasks (BTs) are open-ended, performance-based assessments. BTs are based on a mastery-model of assessment and evaluated on a pass-fail basis. You may use BTs in a variety of ways, from supporting the scoring of a final project, to individual lab assignments. See Introduction to Bridge Task (17.1) for a more in-depth explanation to Bridge Tasks. The main features of a bridge task (17.2) section gives the differences between BTs and other assessments.

An author creates a bridge task either by writing the XML code or by using the edit mode and publishing it. A course coordinator must then place the Bridge Task resource in his/her course's document list. The section on Bridge Task Creation (17.3) describes how to author as well as set up these Bridge Tasks.

Once the bridge task is created and published, the course coordinator must insert the resource in the course's document list (See Setting Up a Bridge Task 17.6). The course coordinator may also create slots to limit the place/time the bridge task may be opened (See Using Slots in Bridge Task 17.6.1). This resource may also be placed inside conditionals resources so that it is accessible only after a particular condition has been met (see Bridge Task and Conditional Resources 17.6.2).

Once the course coordinator has set up the Bridge Task the student is able to open and use the bridge task. A Bridge Task hand-in process using portfolio files may be used by the instructors or students if they wish (See Handing In Bridge Task Files 17.7).

17.1 Introduction to Bridge Task

Bridge tasks consist of one or more individual tasks that describe what the student is to do, usually in the form of a problem to solve. LON-CAPA supports the creation of multiple versions of each task, so that each student may receive a mix of randomly assigned tasks to perform. Students use LON-CAPA to view the bridge task. LON-CAPA supports the

scheduling of BTs to restrict access by IP address range and to allow students to schedule slots of time to take a BT.

When students complete a BT, they upload files they have created for grading. Files are uploaded using LON-CAPA's portfolio system.

LON-CAPA supports the creation of scoring rubrics associated with each individual task to guide graders and ensure inter-rater reliability among multiple graders in a course. Instructors specify criteria to assign an overall score to a BT based on the scores of the individual rubrics.

BTs are used in Michigan State University's CSE 101 course. The course is designed to teach computer competencies by having students solve problems using a variety of computer software (MS-Word, MS-Excel, World Wide Web, and MS Access). In CSE 101, BTs are used for summative assessment, the majority of students' grades are based upon the BTs. Here, students must successfully pass a BT before attempting the next BT. A student's final course grade is based on the highest level BT passed. The CSE 101 class is quite large, approximately 2000 students per semester. Students may take up to one BT per week; typically there are over 14,000 BTs administered per semester. Thus there is a need to quickly and accurately access students. LON-CAPA successfully supports the load requirements.

17.2 Bridge Task Features

There are many ways in which BTs differ from other assessments.

1. **Multiple Versions.** There are multiple versions of a BT. A person taking a BT may receive a different version than the person taking the same BT sitting next to him or her. To create BTs that have different versions, the instructors who created the BT will create multiple sub-questions. The BT engine then chooses one of the sub questions and gives that sub question to the student, thus creating multiple versions.
2. **Essay/task based.** Bridge task questions open-ended. Users create files which they upload and submit to the system.
3. **Rubric-Based Grading.** Each question in a bridge task has scoring rubrics, criteria, associated with them. These criteria are used as the basis of grading. The grading page provides the criteria checklist on which the grader enters whether a student passes or fails each criterion. This ensures inter-rater reliability among multiple graders.
4. **Mandatory and Optional Criteria.** Some criteria can be made mandatory, that is a student will fail if the student does not pass that criteria. Some criteria are optional and the student must pass a certain number of these optional criteria. Criteria are associated with questions, which can also be made optional or mandatory. LON-CAPA calculates whether a student has passed or fail based on the number of mandatory and optional questions the student has passed.
5. **Automatic Bookkeeping.** The system calculates whether the student has passed a BT or not and records it into the database. The system stores
 - a complete record of the BT each student received

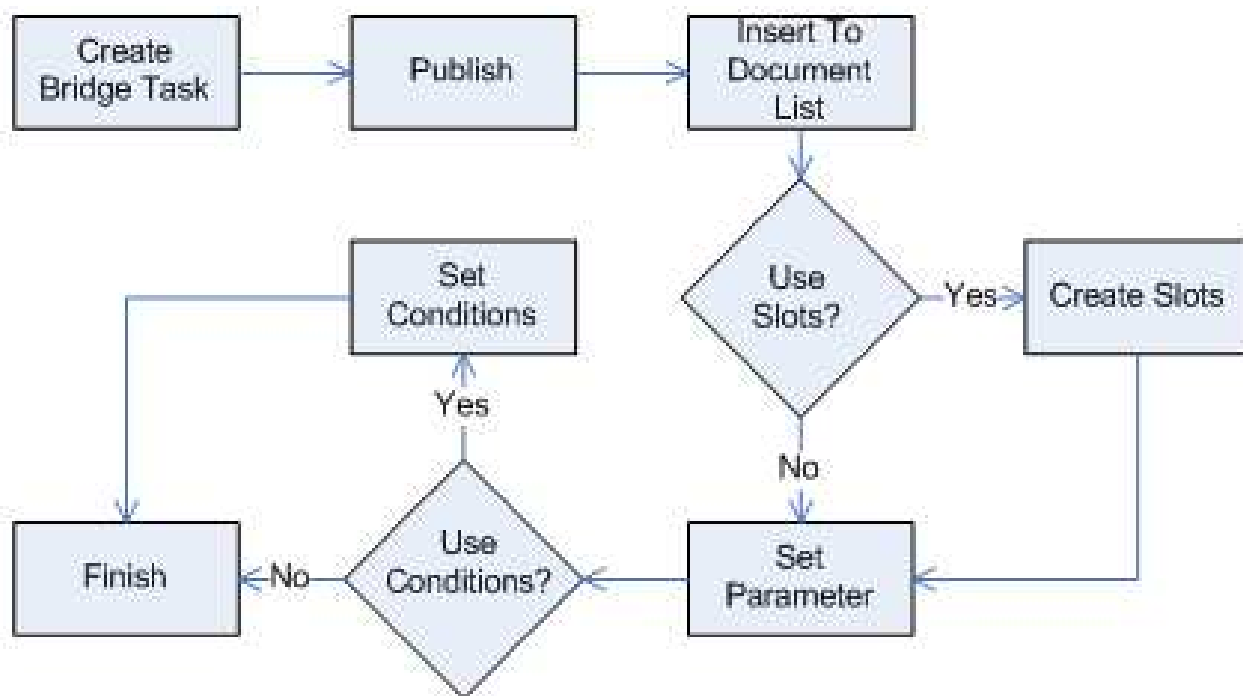


Figure 32: Bridge Task creation flowchart

- when each student's BT was administered
 - the BT instance
 - the files the student turned in
 - the associated grading criteria
 - the grading results (and history of grading, including grader ID)
6. **Sequential Bridge Tasks.** The system can be set in such a way that a student can only take a certain bridge task after passing the previous bridge task on the list. This is done using conditionals in LON-CAPA. There are other ways to use conditionals and bridge tasks to customize the usage of bridge tasks in a course.
 7. **Slots.** Slots can be created to relate students with time and location. This allows control of where and when a student can take a bridge task.
 8. **Proctor Authentication.** Slots also allow a particular proctor to be in a particular location for a bridge task. Each student who is scheduled to take the bridge task must be authenticated by the proctor.

17.3 Authoring Bridge Task

There are multiple steps for creating a bridge task and setting the bridge task up so that the students is able to use it. The flow of the bridge task creation process is shown in figure 32.

There are two ways of creating the bridge task. The first method is to directly edit the XML file being used (See Bridge Task XML Editing 17.4). The second method is to use the LON-CAPA online edit mode (See Bridge Task Mode Editing 17.5).

Once the bridge task is created and published, the course coordinator must insert the resource in the course's document list (See Setting Up Bridge Task 17.6). The course coordinator may also create slots to allow for multiple place/time the bridge task may be opened or to allow for multiple separate attempts at the bridgetask (See Using Slots in Bridge Task 17.6.1). As with any resource, the map it is in may use conditionals to control access to the resources based on other information in the course. (For example, the section of a user, the passing of a prior bridgetask, etc.)

17.4 Bridge Task XML Editing

The LON CAPA .task format is an XML file used directly by LON CAPA. XML is a markup language, much like HTML. A primer on XML is given in section 9. An XML file contains elements in tags, and elements may contain attributes. The syntax, spelling (including capitalization) of the XML file must be exact, otherwise the XML file will generate errors.

The online editor for the .task format is discussed in the next section. This section describes creating the .task file by simply using a text editor such as Notepad.

The .task format consists of four parts:

- The header, which contains information about the file itself. The header can be copied from this documentation as is (see .Task Headers 17.4.1)
- Parameter information. This part consists of the possible values for various parameters in the Bridge Task (see .Task Parameter and Variable 17.4.2)
- Questions section. This part consists of the actual text the user sees as well as all criteria and questions (see .Task Question 17.4.3)
- Footer section. This part consists of the actual text the user sees as well as all criteria and questions (see .Task Finishing Up 17.4.4)

Once all parts are created, the author must publish the file so that it is accessible by course coordinators (see .Task Finishing Up 17.4.4).

17.4.1 .Task Headers

The root node of the .task format is the task element, which is an XML element written as follows:

```
<Task OptionalRequired='0'>
```

The Task element signifies that this is a bridge task. The OptionalRequired is a mandatory attribute, and the value (0 in this case) determines how many optional questions the student must pass.

17.4.2 .Task Parameter and Variable

To create difference versions of Bridge Tasks for each student, parts of the questions are defined using variables. Each variable contains multiple instances of possible values and LON-CAPA randomly selects an instance to give to the student.

All variables are placed inside a Setup element. Each setup element has an id attribute which contains the name of the variable. The possible instances or values of those variables are placed inside the setup element inside an Instance tag. Each Instance element has two mandatory attributes, the OptionalRequired attribute which should be set to 0, and the unique id of that instance. The value of the id attribute can be any text as long as it is unique throughout the document.

The actual data of the instance is placed inside InstanceText tags. Currently the instance data is created with a loncapa/perl script. In this script the parameters of the variable are set. The syntax to set the parameter of a variable is '\$variableName {fieldname} = "fieldValues"'. The variable name is taken from the attribute id from the Setup element, the field name is the name of the parameter the author sets, and the fieldValue is simply the value of the field. The first parameter that must be set is the instance field, with the value being an identifier of the instance.

The example below shows a portion of the bridge task XML file. This portion should be placed inside the task element :Lines between <!-- -- and --> or /* and */ are comments and should not be typed into the editor.

```
<!-- Create a variable named entity Subject -->
<Setup id="entitySubject">

<!-- The first instance. With id instanceHarry -->
<Instance OptionalRequired="0" id="instanceHarry">

<!-- The parameters for this instance -->
<InstanceText>
<script type='loncapa/perl'>
/* The first line must be the instance id */
$entitySubject{instance} = "instanceHarry";
/* The two parameters. Personname = Harry and place=zoo */
$entitySubject{personname} = "Harry";
$entitySubject{place} = "zoo";
</script>

</InstanceText>
</Instance>
<!--End of instanceHarry-->

<!-- The second instance. With id instanceBetty -->
<Instance OptionalRequired="0" id="instanceBetty">
```

```

<!-- The parameters for this instance -->
<InstanceText>
<script type='loncapa/perl'>
/* The first line must be the instance id */
$entitySubject{instance} = "instanceBetty";
/* The two parameters. Personname = Betty and place=park */ $entitySubject{personname} =
$entitySubject{place} = "park";
</script>
</InstanceText>
</Instance>
<!--End of instanceBetty-->

</Setup>

```

The example above describes a variable question. It has two different possible values for the entity "subject", Harry and zoo or Betty and park. Variables can be placed inside the questions by using the variable name and field name. The first line `<Setup id="entitySubject">` creates a variable named `entitySubject` (based on the `id` attribute of this line).

The first instance of this variable is shown from lines 2 to 10. Line two `<Instance OptionalRequired="0" id="instanceHarry">` marks the beginning of the instance element named `instanceHarry` (based on the `id` attribute). The `OptionalRequired` is given a value of 0. Lines 3-9 determine the actual value of this variable. Lines 3 and 4 must be typed as shown. Lines 5-7 define the instance properties, which must be of the form `$ <variable_name> {<property_name> } = <value>`. The first line of the property must be the instance property (see line 5 of example), with the value being the `id` of the instance. Other lines (6-7) can be used for any attributes you wish to define. The closing `</script>`, `</InstanceText>` and `</Instance>` tags must be typed as shown.

Line 12-21 shows the second instance with the same rules as the first instance. Line 23 `</Setup>` gives the closing Setup tag which must be as shown.. The example of the usage of this variable inside the question is this text:

This is a test question. `$entitySubject{personname}` went to the `$entitySubject{place}`.

The LON-CAPA engine will replace any instance of `$<variable_name> (<property_name>)` with the correct value, depending on the randomly chosen instance.

Based on this code, two different questions are possible:

1. This is a test question. Harry went to the zoo
2. This is a test question. Betty went to the park

17.4.3 .Task Questions and Criteria

The task description should be divided into questions. Questions can also be divided into sub-questions. A question or sub-question must have one or more criteria that are the scoring rubrics used to evaluate that question. A task may also have a criteria. Graders use these

criteria to evaluate student work Both questions and criteria are interspersed within the task description, placed where students see them when reviewing their graded bridge task.

Questions are created by using the Question tag. Each question must have a unique id attribute which identifies the question, the value of the id attribute is any text that is unique in the document. Each question must also have the Mandatory attribute which can be set to "Y" if the question is mandatory or "N" otherwise. Finally the question may also have a OptionalRequired attribute, which determines how many optional criteria students must pass to pass the question.

The question element will have the actual text of the question. The questions are created inside the QuestionText element. The question descriptions are placed inside the file by simply typing the text. The text can be marked up to have various formatting. The mark up language used is simple HTML.

Criteria are created using the Criteria tag. The attributes for the criteria tag are similar to the attributes for the questions tag. Criteria tags have id attributes as well as Mandatory attributes. Like for the question tag, the value of the id attribute is any text that is unique in the document, and the Mandatory attribute (values "Y" or "N") determines whether the criteria is mandatory or not. Criteria tags do not have OptionalRequired attributes. The criteria description is created inside the CriteriaText element and can be formatted the same way as the formatting of the questions (using HTML).

The following is an example of a question and criteria element. Question elements are placed inside the task element or inside other question elements. Text between `<!--` and `-->` are comments and should not be typed into the editor.

```
<!-- Beginning of question element with id q.testquestion. A mandatory question where the
<Question id='q.testquestion' Mandatory='Y' OptionalRequired='1'>
<!--The text of the question -->
<QuestionText>
Some test explanation
<!--A mandatory criteria with the id of 'criteria.overall.handin'. This criteria will no
<Criteria id='criteria.overall.handin' Mandatory='Y'>
<!--The text of the criteria-->
<CriteriaText>
Criteria text 1
<!--Grader Notes are not shown to the students at all -->
<GraderNote>
This part cannot be seen by students but is used to give additional info to the grader t
</GraderNote>
</CriteriaText>
</Criteria>
<!--End of the criteria -->

<!--An optional question named q.question1 -->
<Question Mandatory='N' id='q.question1' OptionalRequired='0'>
<QuestionText>
```

The actual test question

```
<!--A mandatory criteria -->
<Criteria id='criteria.question1' Mandatory='Y'>
<CriteriaText>
Criteria text 2
</CriteriaText>
</Criteria>
</QuestionText>
</Question>
<!--End of question q.question1-->
```

```
<!--Some more question text -->
Instructions to the student on submitting files
</QuestionText>
</Question>
<!--End of question q.testquestion-->
```

The question that the students will see is:

"Some test explanation

The actual test question

Instructions to the student on submitting files

"

Any text inside a QuestionText element (lines 2, 14) and not inside a criteria element will be shown to the users. This example has one question, and one optional sub-question. The question element is given in line 1, the id of the question being 'q.testquestion', it is a mandatory question (Mandatory='Y') and it requires that 1 optional question/criteria be passed (OptionalRequired='1'). The sub-question is given in line 13, named 'q.question1', is not mandatory and has no optional questions or criteria that is required to be correctly answered.

The example above shows a code for the creation of criteria. There are 2 criteria in the above question (line 4 and 16). Each criteria has a name (criteria.question1 and criteria.overall.handin) which are both mandatory (the attribute Mandatory is set to 'Y'). The text of the criteria is given inside the CriteriaText element inside each CriteriaElement. The student will never see the criteria when the student is taking the test. The grader will see this on his/her screen:

"Criteria text 1. Grader Note: This part cannot be seen by the students but is used to give additional info to the grader to help evaluating the criteria.

Pass Fail

Comment:

Criteria text 2.

Pass Fail

Comment:

”

When the bridge task is graded the student will see both the questions and the criteria in his space. Anything inside the GraderNote element (line 7) is not shown to the student.

”Some test explanation

Criteria text 1

Pass

The actual test question

Criteria text 2.

Pass

Instructions to the student on submitting files

”

17.4.4 .Task Finishing Up

Once all questions are written, the file must be closed with the `</Task>` tag. The file must be saved with a `.task` extension then placed into a LON CAPA author's folder. To upload the file simply go to the authoring space and simply upload the new file using the upload new document button. Once the file is uploaded to LON CAPA, the author can publish the file so that the domain coordinator will be able to use the file. To do this select publish (or re-publish) in the select action combo box next to the filename.

The LON CAPA interface allows you to view the bridge task from different point of views. You can see it from the student's view, the grader view, or the student's view after feedback:

1. From the list of files authored, click on the bridge task that has just been created/edited. A pull-down menu with the label "Problem Status" is located just above the question
2. Select "Answerable" to see the original view, "Criteria Grading" to see a grader view, and "Show Feedback" to see the feedback view. To see the feedback view properly, you must have graded the student with the grader view. Once you have graded using the feedback view however, the answerable view will be similar to the feedback view.
3. To reset the view a "Reset Submissions" button is provided.

17.5 Bridge Task Edit Mode

Another way of creating Bridge Task is via the edit mode. The edit mode allows authors to create resources (including Bridge Tasks) online. The basic idea of the this editor is the author inserts and deletes different section types into the file to build the bridge task.

If a task file has already been created, click on the filename followed by the "Edit" button to open the file in the colorful editor.

If a task file needs to be created:

1. Enter the filename in the area "Create a new directory or LON-CAPA document" with the combo box at the left of this set to "New File". The filename must have a .task extension.
2. At the second screen, press continue to create a new task. At the third screen, select the blank task template and press the "Create task" button to create this blank task.
3. To go into the the edit mode, press the "Edit" Button.

The edit mode allows you to insert appropriate sections of a document at certain predetermined places. For example, in a bridge task document, you can insert a question section, a criteria section, and an introductory section. To insert a section, find a drop down box with the label "Insert", and select the section to insert. Then click on the "Submit Changes and Edit" button. The section you selected will be inserted in place of the pull down menu.

Additional insert pull-downs will show up allowing you to insert sections before, inside, and after other sections. The choices on the pull down menu may differ depending on the location of the insert pull-down. For example, the insert pull down inside a question section contains question information; this option is not available anywhere else.

In the edit mode, you will see is an insert button and a optional task button. You will need to enter the number of optional tasks/questions that the student need to answer correctly to pass the bridge task. You can leave this field empty until after you have finished writing the whole bridge task.

This document discusses important sections needed to create a bridge task. These sections can be created in any order.

1. The introductory and closing information 17.5.1
2. Variables 17.5.3
3. Question and criteria 17.5.2

Once the document has been created, the author must publish the bridge task so it becomes available (See Edit Mode Finishing Up 17.5.4)

17.5.1 Introductions

The introductory information and closing information are just text that are shown to the students but are not part of any questions. The introductory information should be placed at the beginning of the bridge task and the closing information should be placed at the end of the bridge task.

To insert these sections:

1. Select "Introductory Information" or "Closing Information" in the insert pull-down menu and press the submit button.
2. A new section appears with a text box inside it. You may enter any text inside the text box and the text may be formatted with HTML tags. Everything inside the text box will be shown to the students as introductory or closing text.

Required number of passed optional elements to pass the Task:

Insert: 1

Question Delete: Insert: 2

3 Id: q1 Passing is Mandatory: Yes 4

Required number of passed optional elements to pass the Question: 5

Question Information Delete: Insert:

Text Block Delete: 6

Check Spelling

Insert:

Insert:

Insert:

Figure 33: Bridge Task question creation screenshot

17.5.2 Questions and Criteria

The task description should be divided into questions. Questions can also be divided into sub-questions. A question or sub-question must have one or more criteria that are the scoring rubrics used to evaluate that question. A task may also have a criteria. Graders use these criteria to evaluate student work. Both questions and criteria are interspersed within the task description, placed where students see them when reviewing their graded bridge task.

Figure 33 shows a screenshot of the question creation page. To create a question:

1. Select "Question" in the main insert pull-down (circled as 1).
2. Press the submit changes and edit button.
3. In this question create a unique id which can be any kind of text (in circle 3).
4. Set whether a student must pass this question by setting the value of "Passing is Mandatory" to "yes" (circle 4) or whether this question is optional by setting the value to "no".
5. Set the number of optional subquestions/criteria that the students must get correct in order to pass this question. You can also wait until later to fill this in.

Figure 34: Bridge Task criteria creation screenshot

6. Create the question information by choosing Question information in the insert pull-down (circled as 2) and pressing the submit button.
7. A text box should appear (circle 6) and you can insert any text in this text box. Anything that is typed in this text box will appear to the student as part of the question. HTML tags can also be used to format the text in the text box. Subquestions can be added by inserting a new question (circle 2).

To add a criteria (see Figure 34):

1. Instead of choosing "Question" in the insert pull down, select "Question Criteria" (circled as 1). Pressing the submit button will bring up the Question criteria as well as the criteria information where you can write the criteria (Circled 3). The criteria is shown to the student only when the student is reading his/her feedback. The criteria is also shown to the grader in order for the grader to be able to grade the student submission.
2. In the criteria block, first create a unique id for this criteria (any text) (circle 1).
3. Specify whether a criteria is mandatory or not, this is set in the Passing is mandatory pull-down similar to that of the question (circle 2).

Figure 35: Bridge Task variable creation screenshot

4. Text in the text block (circle 4) is shown as the criteria. The text can be formatted with HTML.
5. You can also insert a grader note which will only be shown to the user. To do this select "Text to display to grader" in the insert pull down menu of circle number 3.

17.5.3 Parameter and Variable

To create difference versions of Bridge Tasks for each student, parts of the questions are defined using variables. Each variable contains multiple instances of possible values and LON-CAPA randomly selects an instance to give to the student. Figure 35 shows a screenshot of this process.

To create a variable:

1. Choose "Setup ..." in the insert pull-down menu (circled 1 in figure).
2. Press the Submit Changes and Edit just above the work space.
3. A new box should appear with the label "Setup ...". In this box, fill out the id box (circled 3) with any text that is unique to the document. This id is the name of the variable and will be used when creating the values for the variable.

4. The setup box has an insert pull-down menu next to the label (circled 2), select "Specific Question Instance" in this pull down menu, then again press the Submit Changes and Edit button.
5. This creates one single instance of a set of possible values. For each instance created, a new "Specific Question Instance" must be created.
6. Right now a box should appear inside the "Setup... " box with the label "Specific Question Instance". Insert a unique id for that instance which can be any unique text (circled 4). This id is the instance name and is used as one of the property of the variable.
7. In the question instance block, select "Information for the Instance" in the insert pull down (circled 5). Again press submit button.
8. Add a new script (circled 6) in the insert pull down. A new text block should appear.
9. In this text box, a perl script will be created (circled 7). A set of parameters for this variable is added. The syntax to set the parameter of a variable is '\$variableName {fieldname} = "fieldValue"'. The variable name is taken from the id field of the Setup block, the field name is the name of the parameter the author sets, and the fieldValue is simply the value of the field. The first parameter that must be set is the instance field, with the value being an identifier of the instance (which is the id of the specific question instance block).

The example below shows two instances of this script for the variable entitySubject with two instances, 'instanceHarry' and 'instanceBetty'.

The first instance (instanceHarry) is:

```
$entitySubject{instance} = "instanceHarry";
$entitySubject{personname} = "Harry";
$entitySubject{place} = "zoo";
```

The second instance (instanceBetty) is:

```
$entitySubject{instance} = "instanceBetty";
$entitySubject{personname} = "Betty";
$entitySubject{place} = "park";
```

The example above describes a variable question. It has two different possible values for the entity "subject", Harry and zoo or Betty and park. Variables can be placed inside the questions by using the variable name and field name. The example of the usage of this variable inside the question is this text:

This is a test question. \$entitySubject{personname} went to the \$entitySubject{place}.

The LON-CAPA engine will replace any instance of \$<variable_name> (<property_name>) with the correct value, depending on the randomly chosen instance.

Based on this code, two different questions are possible:

1. This is a test question. Harry went to the zoo
2. This is a test question. Betty went to the park

17.5.4 Edit Mode Finishing Up

Once the construction is complete, click on the "Submit Changes and View" button. Click on list to list the current author's directory. The author should now publish the file so that the domain coordinator will be able to use the file. To do this select publish (or re-publish) in the select action combo box next to the filename.

The LON CAPA interface allows you to view the bridge task from different point of views. You can see it from the student's view, the grader view, or the student's view after feedback.

1. From the list of files authored, click on the bridge task that has just been created/edited.
2. A pull-down menu with the label "Problem Status" is located just above the question. select "Answerable" to see the original view, "Criteria Grading" to see a grader view, and "Show Feedback" to see the feedback view.
3. To see the feedback view properly, you must have graded the student with the grader view.
4. Once you have graded using the feedback view however, the answerable view will be similar to the feedback view. To reset the view a "Reset Submissions" button is provided.

17.6 Setting Up a Bridge Task

The first step in making bridge tasks available to students is to include it in the document space. To do this:

1. Enter the course coordinator space for the course
2. Go to "Course Documents" and click on import. This page gives the list of files that you have created.
3. Select the bridge task and press import to insert the file to the list of documents.
4. After importing the document, you must re-initialize the course.

Now the assignment is in the list of documents, but it is not available for students. There are two ways of making bridge tasks available to students. One method is by using slots (the default method), which restrict the bridge task document to open at certain time/place only. The other method allows students to take bridge task like any other Lon-CAPA assignments, that is at any time they want within opening and closing dates. This method does not restrict the access to particular computers or rooms. Any student in the course may open the BT.

If you want to use slots, first create the slots (See Bridge Task and Slots 17.6.1). Once the BT is imported (and slots are created) go back to the document list and click on the

bridge task resource, this should take you to a page that shows the resource content. If you are not using slots, click on the bridge task resource. Whether using or not using slots, click on PPRM (a button on the top menu) to modify parameters for this resource.

Set the opening date and the due date (if any) in the for resource column. To do this, click on the * in the in course for resource column. A pop up should appear. Enter the date you want the resource to be available and click the store link.

If you are using slots:

1. Change the "Use slot based access controls" parameter to "Yes".
2. Change the "Slots of availability" parameter for the course in for resource column to the name of the slot that you created. You may need to change the input type (the combo box at the top of the popup) to 'String Value' instead of 'default'.

If you are not using slots:

1. Change the "Use slot based access controls" parameter to "no" by clicking on the * in the for resource column

The bridge task should now be available to students. To see the BT in the student view, switch to a student role in your course. Navigate contents and click on the resource.

17.6.1 Bridge Task and Slots

To restrict when and where BTs can be taken, the slots feature of LON-CAPA is used. Slots allow the instructors to specify the room, the time, the students that can take the Bridge Tasks, and finally any proctors that will authenticate the students.

17.6.2 Bridge Task and Conditional Resources

It is possible to configure bridge tasks such that only when a student passes a Bridge Task does the next Bridge Task appear to the student. This feature is useful in such cases as Bridge Tasks are related to one another, and the results of one Bridge Task is needed for the next Bridge Task.

To create configure bridge tasks these way, a sequence page must be created. A resource author starts at his/her authoring space (Main menu: AUTH). Once in the authoring space, the author create a new assembled sequence. See the section 14 for a more concise reference on condition and sequences.

The author must create various conditionals to determine which resources to show. The easiest way to do this is by using advanced edit. The advanced edit visualizes the various conditions and resources. This interface needs popups, so make sure popups are enabled in the browser.

In the main starting view of the advanced edit, there are two squares, marked start and end respectively. Start by creating a link between the start square and the end square. To do this click on the start square (on the row under the text start), select "Link Resource", then click on the end square. A link is now created between the start and the end boxes.

Once a link is created the author adds resources and conditions. Click on the link and then choose insert resource into link. This allows the placement of a resource such as a bridge task into the sequence. Once the resource is placed into the link, the author should create a mapalias. To do this, select the resource, then click on set parameters, and change the value of Custom Parameter to an alias.

Under each square is an area of the same color as the square, these areas can be used to create conditionals to get to the next resource to the sequence. This document only discusses using conditionals to block access to the next resource (including Bridge Tasks).

Add conditionals by clicking on the area just under the squares. A popup window will appear with a condition box and some options. The condition box allows the author to enter a variety of different conditions. To block access to the next resource, select the second option "Blocking this link if false".

The conditions used will be :

```
&EXT('user.resource.resource.0.awarded','<alias of resource>') eq '1'
```

<alias of resource> should be changed to the alias set for the previous resource that the student needs to pass before the next resource is unblocked. If the student has not passed a particular homework/problem/bridge task, the value of user.resource.resource.0.awarded is 0 otherwise the value is 1. This line checks whether the value of user.resource.resource.0.awarded for that resource is equals to 1, if not the link is blocked and the next resources are not shown to the user.

17.7 Handing In Bridge Task Files

After a student finishes creating the Bridge Task, the student must hand in the files. There are many ways to do this, students can email the files necessary, or students can email a link to the instructor that shows where his files are. LON-CAPA provides a way to upload file using the portfolio. The portfolio is a space given to each student which can contain files and is shareable.

The process of handing in files consists of:

1. Upload files into portfolio
2. Select the files to be submitted
3. Submit the files

At the end of every bridge task there will be a box to submit files in the portfolio for grading. When the student click on the link "select portfolio files", a new window is opened showing the content of the student's portfolio. The student can create directories in the portfolio and upload files using the upload file buttons.

Once the files that are needed are uploaded, the student select (by checking the checkbox next to the filenames) of all files that he needs to submit to the instructor. Once all files needed are checked off, the student needs to press the "select checked files and close window" button.

By pressing the close window button, the student's portfolio window should close, and the student should see his or her Bridge Task Page. The text field in the upload files into portfolio should contain the list of names. The student must then press 'submit answer' to submit the answer to be graded.

If the student needs to resubmit the file, the student must repeat the process and check off all the files (not only new files) that needs to be submitted. Once a file is submitted, the student may not overwrite or delete the files.

18 Appendix: Symbols in TeX

18.1 Greek Symbols

If you are viewing this online, copy and paste the text from any of the right columns into your text area to get the symbol on the left.

Symbol	HTML character entities	Copy this column
α	<code>&alpha;</code> or <code>&#945;</code>	<code><m>\$\alpha\$</m></code>
β	<code>&beta;</code> or <code>&#946;</code>	<code><m>\$\beta\$</m></code>
γ	<code>&gamma;</code> or <code>&#947;</code>	<code><m>\$\gamma\$</m></code>
Γ	<code>&Gamma;</code> or <code>&#915;</code>	<code><m>\$\Gamma\$</m></code>
δ	<code>&delta;</code> or <code>&#948;</code>	<code><m>\$\delta\$</m></code>
Δ	<code>&Delta;</code> or <code>&#916;</code>	<code><m>\$\Delta\$</m></code>
ϵ	<code>&epsilon;</code> or <code>&#949;</code>	<code><m>\$\epsilon\$</m></code>
ε		<code><m>\$\varepsilon\$</m></code>
ζ	<code>&zeta;</code> or <code>&#950;</code>	<code><m>\$\zeta\$</m></code>
η	<code>&eta;</code> or <code>&#951;</code>	<code><m>\$\eta\$</m></code>
θ	<code>&theta;</code> or <code>&#952;</code>	<code><m>\$\theta\$</m></code>
ϑ	<code>&thetasym;</code> or <code>&#977;</code>	<code><m>\$\vartheta\$</m></code>
Θ	<code>&Theta;</code> or <code>&#920;</code>	<code><m>\$\Theta\$</m></code>
ι	<code>&iota;</code> or <code>&#953;</code>	<code><m>\$\iota\$</m></code>
κ	<code>&kappa;</code> or <code>&#954;</code>	<code><m>\$\kappa\$</m></code>
λ	<code>&lambda;</code> or <code>&#955;</code>	<code><m>\$\lambda\$</m></code>
Λ	<code>&Lambda;</code> or <code>&#923;</code>	<code><m>\$\Lambda\$</m></code>
μ	<code>&mu;</code> or <code>&#956;</code>	<code><m>\$\mu\$</m></code>
ν	<code>&nu;</code> or <code>&#957;</code>	<code><m>\$\nu\$</m></code>
ξ	<code>&xi;</code> or <code>&#958;</code>	<code><m>\$\xi\$</m></code>
Ξ	<code>&Xi;</code> or <code>&#926;</code>	<code><m>\$\Xi\$</m></code>
π	<code>&pi;</code> or <code>&#960;</code>	<code><m>\$\pi\$</m></code>
ϖ	<code>&piv;</code> or <code>&#982;</code>	<code><m>\$\varpi\$</m></code>
Π	<code>&Pi;</code> or <code>&#928;</code>	<code><m>\$\Pi\$</m></code>
σ	<code>&sigma;</code> or <code>&#963;</code>	<code><m>\$\sigma\$</m></code>
ς		<code><m>\$\varsigma\$</m></code>
Σ	<code>&Sigma;</code> or <code>&#931;</code>	<code><m>\$\Sigma\$</m></code>
τ	<code>&tau;</code> or <code>&#964;</code>	<code><m>\$\tau\$</m></code>
υ	<code>&upsilon;</code> or <code>&#965;</code>	<code><m>\$\upsilon\$</m></code>
Υ	<code>&Upsilon;</code> or <code>&#933;</code>	<code><m>\$\Upsilon\$</m></code>
ϕ	<code>&phi;</code> or <code>&#966;</code>	<code><m>\$\phi\$</m></code>
φ		<code><m>\$\varphi\$</m></code>
Φ	<code>&Phi;</code> or <code>&#934;</code>	<code><m>\$\Phi\$</m></code>
χ	<code>&chi;</code> or <code>&#967;</code>	<code><m>\$\chi\$</m></code>
ψ	<code>&Psi;</code> or <code>&#968;</code>	<code><m>\$\psi\$</m></code>
Ψ	<code>&Psi;</code> or <code>&#936;</code>	<code><m>\$\Psi\$</m></code>
ω	<code>&omega;</code> or <code>&#969;</code>	<code><m>\$\omega\$</m></code>
Ω	<code>&Omega;</code> or <code>&#937;</code>	<code><m>\$\Omega\$</m></code>
ρ	<code>&rho;</code> or <code>&#961;</code>	<code><m>\$\rho\$</m></code>
ϱ		<code><m>\$\varrho\$</m></code>

18.2 Other Symbols

If you are viewing this online, copy and paste the text on any of the right columns into your text area to get the symbol on the left.

Symbol	HTML entity	Copy this column
\pm	<code>&plusmn;</code> or <code>&#177;</code>	<code><m>\$\pm\$</m></code>
\times	<code>&times;</code> or <code>&#215;</code>	<code><m>\$\times\$</m></code>
\div	<code>&divide;</code> or <code>&#247;</code>	<code><m>\$\div\$</m></code>
\cdot	<code>&middot;</code> or <code>&#183;</code>	<code><m>\$\cdot\$</m></code>
\star		<code><m>\$\star\$</m></code>
\circ	<code>&deg;</code> or <code>&#176;</code>	
\bullet	<code>&#149;</code>	<code><m>\$\bullet\$</m></code>
\dagger		<code><m>\$\dagger\$</m></code>
\ddagger		<code><m>\$\ddagger\$</m></code>
\dagger	<code>&#134;</code>	<code><m>\$\dagger\$</m></code>
\ddagger	<code>&#135;</code>	<code><m>\$\ddagger\$</m></code>
\copyright	<code>&copy;</code> or <code>&#169;</code>	<code><m>\$\copyright\$</m></code>
\leq	<code>&le;</code> or <code>&#8804;</code>	<code><m>\$\leq\$</m></code>
\geq	<code>&ge;</code> or <code>&#8805;</code>	<code><m>\$\geq\$</m></code>
\neq		<code><m>\$\neq\$</m></code>
\ll		<code><m>\$\ll\$</m></code>
\gg		<code><m>\$\gg\$</m></code>
\simeq		<code><m>\$\simeq\$</m></code>
\perp	<code>&perp;</code> or <code>&#8869;</code>	<code><m>\$\perp\$</m></code>
\parallel		<code><m>\$\parallel\$</m></code>
\leftarrow	<code>&larr;</code> or <code>&#8592;</code>	<code><m>\$\leftarrow\$</m></code>
\Leftarrow	<code>&lArr;</code> or <code>&#8656;</code>	<code><m>\$\Leftarrow\$</m></code>
\rightarrow	<code>&rarr;</code> or <code>&#8594;</code>	<code><m>\$\rightarrow\$</m></code>
\Rightarrow	<code>&rArr;</code> or <code>&#8658;</code>	<code><m>\$\Rightarrow\$</m></code>
\uparrow	<code>&uarr;</code> or <code>&#8593;</code>	<code><m>\$\uparrow\$</m></code>
\Uparrow	<code>&uArr;</code> or <code>&#8657;</code>	<code><m>\$\Uparrow\$</m></code>
\leftrightarrow	<code>&harr;</code> or <code>&#8596;</code>	<code><m>\$\leftrightarrow\$</m></code>
\Leftrightarrow	<code>&hArr;</code> or <code>&#8660;</code>	<code><m>\$\Leftrightarrow\$</m></code>
$\sqrt{}$	<code>&radic;</code> or <code>&#8730;</code>	<code><m>\$\sqrt{}\$</m></code>
∂	<code>&part;</code> or <code>&#8706;</code>	<code><m>\$\partial\$</m></code>
\sum	<code>&sum;</code> or <code>&#8721;</code>	<code><m>\$\sum\$</m></code>
\int	<code>&int;</code> or <code>&#8747;</code>	<code><m>\$\int\$</m></code>
∞	<code>&infin;</code> or <code>&#8734;</code>	<code><m>\$\infty\$</m></code>

19 Appendix: Physical Units

Physical Units Accepted by LON-CAPA

The following subsections show the physical units that LON-CAPA accepts. The symbols must be used when entering the units, for example “35 kg”.

Note that compound units are formed by using *, / and ^. For example, an acceleration might be in terms of “m/s²” or meters per second squared. This could also be expressed as “m/s/s”. Units of Newton-meters (for torque) would be entered as “N*m”. Parentheses may be used to guarantee the correct sense of the unit. Kilometers per Ampere-hour could be written as “km/(A*hr)” or “km/A/hr” but not “km/A*hr”. The last option would be interpreted as kilometer times hours per Ampere.

LON-CAPA will automatically perform some conversions between units of the same dimension when units are provided for a problem. You can provide an answer of “1.45 km” for a distance. If the computer expects the answer in cm, it will convert your answer before comparing against the numerical solution.

Please note that if your units are inappropriate, the computer has no way of checking the appropriateness of your answer. If units are required, only once appropriate units are provided will the system check your numerical answer.

Base Units

#	name	symbol	comment
	meter	m	# length
	kilogram	kg	# mass
	second	s	# time
	ampere	A	# electric current
	kelvin	K	# thermodynamic temperature
	mole	mol	# amount of substance
	candela	cd	# luminous intensity
	decibel	dB	# log of pressure amplitude

Prefixes

#	Prefix	symbol	factor
	yotta	Y	10 ^{24}
	zetta	Z	10 ^{21}
	exa	E	10 ^{18}
	peta	P	10 ^{15}
	tera	T	10 ^{12}
	giga	G	10 ⁹
	mega	M	10 ⁶
	kilo	k	10 ³

hecto	h	10^2
deca	D	10^1
deci	d	10^{-1}
centi	c	10^{-2}
milli	m	10^{-3}
micro	u	10^{-6}
nano	n	10^{-9}
pico	p	10^{-12}
femto	f	10^{-15}
atto	a	10^{-18}
zepto	z	10^{-21}
yocto	y	10^{-24}

Derived Units

Derived Unit

# name	symbol	comment
gram	g	# mass
minute	min	# time
hour	hr	# time
hour	h	# time
day	day	# time
day	days	# time
year	yr	# time
pound	lb	# mass
ounce	oz	# mass
inch	in	# length
foot	ft	# length
mile	mi	# length
yard	yd	# length
nautical_mile	n_mi	# length, nautical mile (UK)
rood	rood	# area, rood
acre	acre	# area, acre
hertz	Hz	# frequency
litre	L	# volume
newton	N	# force
pound_force	lbf	# force
dyne	dyn	# force
pascal	Pa	# pressure, stress
bar	bar	# pressure
mmHg	mmHg	# pressure, millimeter of mercury
torr	torr	# pressure
atm	atm	# standard atmosphere

joule	J	# energy, work, heat
electronvolt	eV	# energy
calorie	cal	# energy
Btu	Btu	# energy
watt	W	# power, radiant flux
coulomb	C	# electric charge
volt	V	# electric potential, electromotive force
ohm	ohm	# electric resistance, use this in /ANS
ohm	ohms	# electric resistance
ohm	Ohm	# electric resistance
ohm	Ohms	# electric resistance
mho	mho	# electric conductance
mho	mhos	# electric conductance
mho	Mho	# electric conductance
mho	Mhos	# electric conductance
siemens	S	# electric conductance
farad	F	# electric capacitance
tesla	T	# magnetic flux density
weber	Wb	# magnetic flux
henry	H	# inductance
radian	rad	# plane angle
degree	deg	# plane angle (Pi rad=180 deg)
steradian	sr	# solid angle
lumen	lm	# luminous flux
lux	lx	# illuminance
becquerel	Bq	# activity (radioactive)
curie	Ci	# activity (radioactive)
gray	Gy	# absorbed dose (of radiation)
sievert	Sv	# dose equivalent (dose equivalent index)
astroUnit	AU	# mean distance earth to sun
celcius	degC	# multiplicatively OK
fahrenheit degF		# multiplicatively OK
molarity	M	# chemisty
amu	amu	# atomic mass unit
amu	u	# atomic mass unit
lightspeed	c	# speed of light
cubiccentimeter	cc	# cubic centimeter
electroncharge	e	# elementary charge
hbar	hbar	# Planck constant
milesperhour	mph	# miles per hour
rpm	rpm	# rounds per minute
rpm	rpms	# rounds per minute
parsec	pc	# parsec

Interpretation

The coded units are interpreted in the order of basic unit, derived unit, then prefix. For example, “T” will be matched against “tesla” instead of considered the prefix “T”. Another example is that “min” will match “minutes” instead of treated as a combination of the prefix “m” and units.

Index

- *response, 68
- .css, 10
- .htm, .html, xhtm, xhtml, 10
- .page, 11
- .problem, 11
- .sequence, 12
- &EXT, 68, 114

- absolute tolerance, 37
- adaptive hints, 68
- algebra, 97
- answer, 93, 94
- answer display, 28
- awarddetail, 95

- bgimg, 101
- block, 103
- browsing resources, 17
- bugs, 18
- Bugzilla, 18

- cascading style sheet, 10
- chem, 97
- ci, 93
- Clair de Lune, 21
- colorful editor, 24
- comment, 104
- comment markers, 7
- concept groups, 19
- conceptgroup, 96
- conditional hints, 68
- content page, 10, 19
- cs, 93
- css, 10
- customhint, 97

- default, 95
- description, 96, 102
- display, 94, 102
- displayduedate, 98
- displaytitle, 99
- dynamic plot, 27, 50

- endouttext, 104

- eval=on, 100

- foil, 19, 96
- foilgroup, 96
- form, 94
- format algebra, 97
- format number, 98
- format numbers in text block, 41
- format reaction, 97
- Formula Response, 21
- Formula Response, Creating, 42
- formulahint, 96

- gnuplot, 50
- Greek symbols, 23

- height, 101
- help, 17
- help icon, 17
- hintgroup, 96
- hintpart, 96
- hints, 68
- HTML file, 10, 19
- HTML page, 22

- id, 95
- image, 93
- import, 66, 103

- label, 102
- labelgroup, 102
- library, 66
- listserv, 18
- location, 102
- LON-CAPA listserv, 18
- loncapagrade, 95

- m, 100
- map, 11, 12
- mc, 93
- message, 95
- Metadata, 77

- name, 94, 95, 102

- notsolved, 28, 103
- num, 98
- num output tag, 41
- Numerical Response, 21, 35
- numericalhint, 96
- Option Response, 19
- optionhint, 96
- output tags, 23
- page, 11
- parse, 98
- parserlib, 103
- part, 103
- picture, 27
- plot, 50
- plots, 27
- postanswerdate, 28, 46, 103
- preduedate, 28, 103
- problem, 11, 103
- problemtype, 102
- Publishing Resource, 77
- Radio Response, 19, 30
- radiobuttonhint, 97
- random_permutation, 66
- randomizing parts, 67, 103
- randomlabel, 101
- randomlist, 67, 103
- rectangle, 93
- relative tolerance, 37
- relative tolerance, in formula response, 45
- Reset Submissions, 25
- Resource, Publishing, 77
- Sample Points, 45
- script, 39, 103
- script variables, viewing, 68
- scriptlib, 103
- search repository, advanced, 16
- search repository, basic, 15
- Sequence, 86
- sequence, 12
- solved, 28, 103
- spelling checker, 23
- standalone, 98
- startouttext, 104
- statement, 39
- String Response, 21
- stringhint, 96
- strings, 40
- suggestions, 18
- surveys, 27
- symbols, html, 23
- text, 94
- textfield, 97
- textline, 97
- texwidth, 101
- togglebox, 100
- tolerance, 37
- tolerance, absolute, 37
- tolerance, in formula response, 45
- tolerance, relative, 37
- TTH, 100
- type, 93, 94, 96, 102
- url, 94
- value, 102
- variables in text block, 40
- while, 103
- width, 101
- window, 100
- windowlink, 100
- x, 102
- XML editor, 25
- y, 102