Learning *Online* Network with CAPA

Author's Tutorial And Manual

17th August 2005

LON-CAPA Group

Michigan State University

Contents

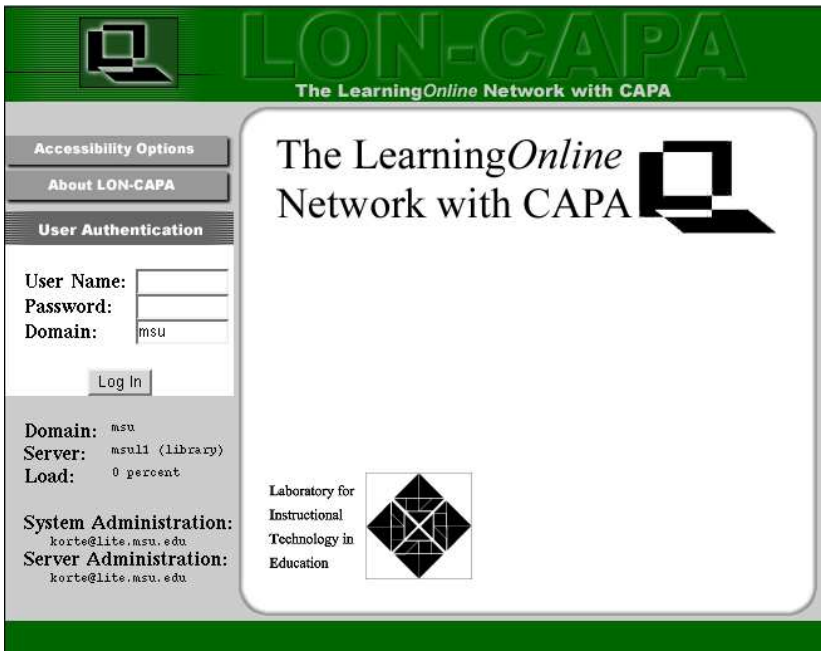


Figure 1: LON-CAPA Log in screen

1 Introduction to LON-CAPA

LON-CAPA is a web-based interface that helps to organize and present your course website, deliver and manage problems, and manage student enrollment. All author functions are done through a web browser (Netscape 4.x or higher, a recent Mozilla, or IE 5+ required).

At this time, you should have:

- developed your objectives for your course.
- developed your problems for input into LON-CAPA and determined the appropriate question formats.

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`.

Much of this document can be used as a tutorial that will introduce you to the authoring system.

For additional help, visit our FAQ at `http://help.loncapa.org/`.

1.2 Login as Course 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.

Figure 2: Author Remote Control

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 Author Remote Control

The Author Remote Control will automatically load whenever you log in to LON-CAPA as the course instructor. The Author Remote Control is a separate window in your browser, and is automatically sized and placed in the upper left of the screen. The Remote Control is a tool that allows you to switch between functions and roles within LON-CAPA.

When you move your mouse over the buttons in the remote, the sixteen gray boxes will show a reminder of what that button does.

- **ROLES (CHOOSE ROLE)** allows you to select which user role to assume for this session.
- **COM (COMMUNICATION)** allows you to access the communication functions in the system.
- **CUSR (USER ROLES)** brings up a page that allows you to create new users and change user privileges.
- **CSTR (CONSTRUCT)** displays the construction space for your account.

- **RES (RESOURCE SPACE)** allows you to browse the LON-CAPA network directory.
- **SRC (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.

2 Creating Content Using LON-CAPA

LON-CAPA provides three types of resources for organizing your course website. LON-CAPA refers to these resources as Content Pages, Problems, and Maps. Maps may be either of two types: Sequences or Pages. You will use these LON-CAPA resources to build the outline, or structure, for the presentation of your course to your students.

- A **Content Page** displays course content. It is essentially a conventional HTML page. These resources use the extension ".html".
- A **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".
- A **Sequence** is a type of **Map** which is used to link other resources together. The users of this resource can use directional buttons on their remote or the NAV button to follow the sequence. Sequences are stored in files that must use the extension ".sequence". Sequences can contain other sequences and pages.
- A **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".

2.1 Description of the Construction Space

The Construction Space is the section of LON-CAPA where you create and manage your course resources. The figure explains what each button does.

2.2 How to Create New Content Pages

Content Pages are HTML documents that display the course information you are presenting.

Many users use tools such as Dreamweaver to create web pages. To upload HTML files generated with such tools, you can use the **Browse** button in the Construction Space, locate your HTML file, and use the **Upload File** button to create a content page in LON-CAPA. Remember to upload any graphics your generated web pages may have included.

To create new Content Pages, do the following:

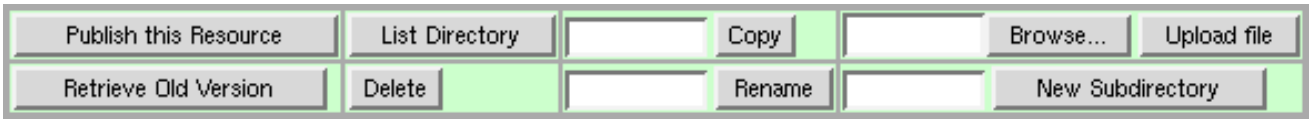


Figure 3: Construction Space

Contents of the Construction Space:

Button Name	Description
Publish this Resource	Opens the Resource Publishing window.
List Directory	Lists the contents of the current working directory
Copy	Type a new name in the entry box to make a copy the current resource
Browse	Helps you select a file to upload
Upload File	Uploads the selected file to your Construction Space
Retrieve Old Version	Load an older version of a resource if you have multiple versions
Delete	Deletes the current resource
Rename	Type a new name in the associated entry box to rename a resource
New Subdirectory	Type a name in the entry box to create a new directory

1. Click the **CSTR** button on the LON-CAPA remote. Your web page will change to your Construction Space.

2. In the Location bar of your browser, type in the full URL of the new Content Page. Make sure the last part of the URL ends with ".html", for example, *http://(your library server)/priv/username/new_resource.html* .
   Press the Return or Enter key.

3. Type the content into the editor, *OR* copy and paste HTML source code obtained through the use of some other HTML authoring program into the editor.

4. Optionally, click the **View** button to preview your Content Page.

5. Finally, click the **Save this** button *OR* click the **Save and then attempt to clean HTML** button.

Repeat this process as many times as necessary to create your Content Pages.

If you're following this as a tutorial, create at least one content page, which we'll use later as raw material. Visit the FAQ at `http://help.lon-capa.org/` if you get "unmatched tag" warnings.

2.3 How to Edit Existing Content Pages

You may edit any Content Pages that have been created.

To edit Content Pages:

1. Click the **CSTR** button on the LON-CAPA Remote. Your web page will change to your Construction Space.

2. Click on the link for the name of the Content Page to edit. The Content Page editor will load and display the current edition of the Content Page.

3. Press the **Edit** button. Edit the HTML code, or copy and paste HTML source code into the editor.

4. Finally, click the **Save this** button *OR* click the **Save and then attempt to clean HTML** button. If you do not do this, your work will not be saved.

Once you've saved your page, you can click the **View** button to preview your Content Page.

2.4 Creating Online Problems Using LON-CAPA

If you're following this as a tutorial, create one of each of these problem types now. We'll be using them later as raw material to assemble maps and sequences.

While several problem types are listed here, in LON-CAPA all problems are actually the same. All problems are written in XML, which can be obtained and edited with the **EditXML** button. The problem types listed in this manual are just templates. As your knowledge advances, you may wish to play with the XML representation directly to see what you can do.

2.5 Problem Types

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

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.

2.6 Foils

In the LON-CAPA system, a **Foil** is the statement after 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.

2.6.1 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.

2.6.2 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.

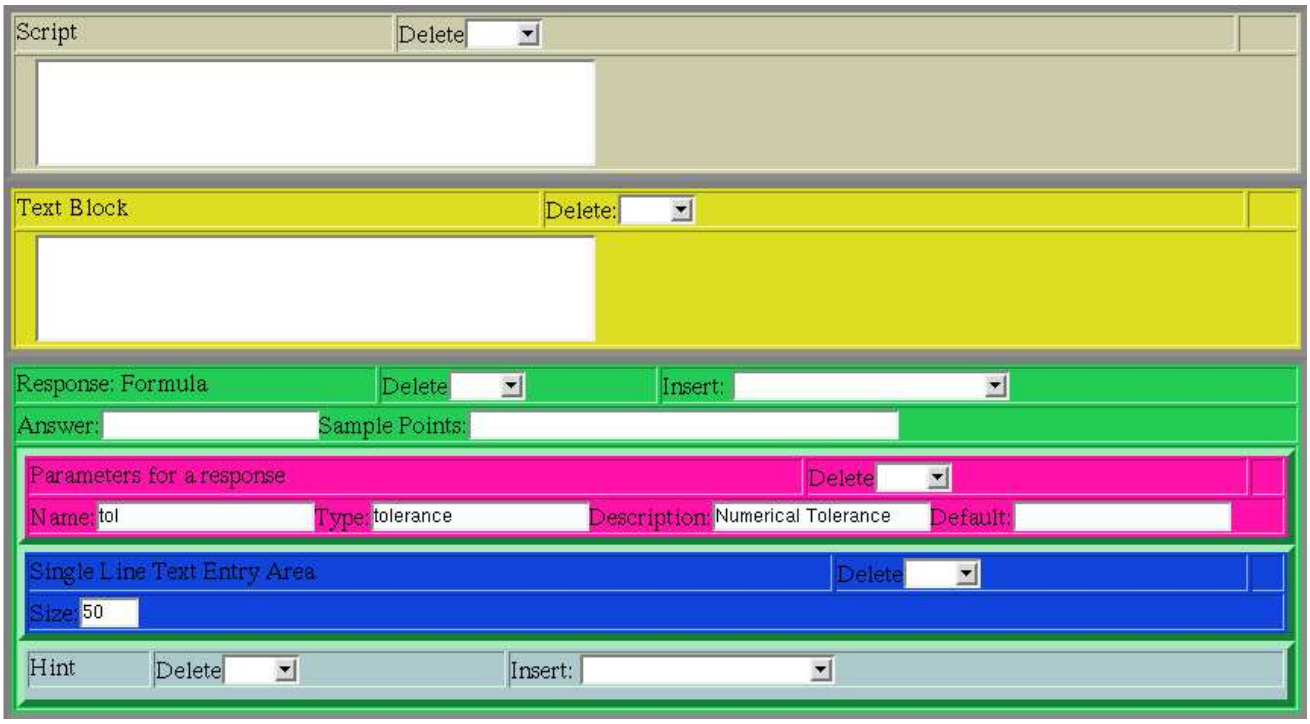


Figure 4: Formula Response Problem

2.6.3 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.

2.6.4 Numerical Response Problems

Numerical Response problems are answered by entering a number and (optionally) a unit, such as 2.5 m/sˆ2. Tolerance and required significant digits can be specified as well.

2.6.5 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*xˆ2 + 14 - 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.

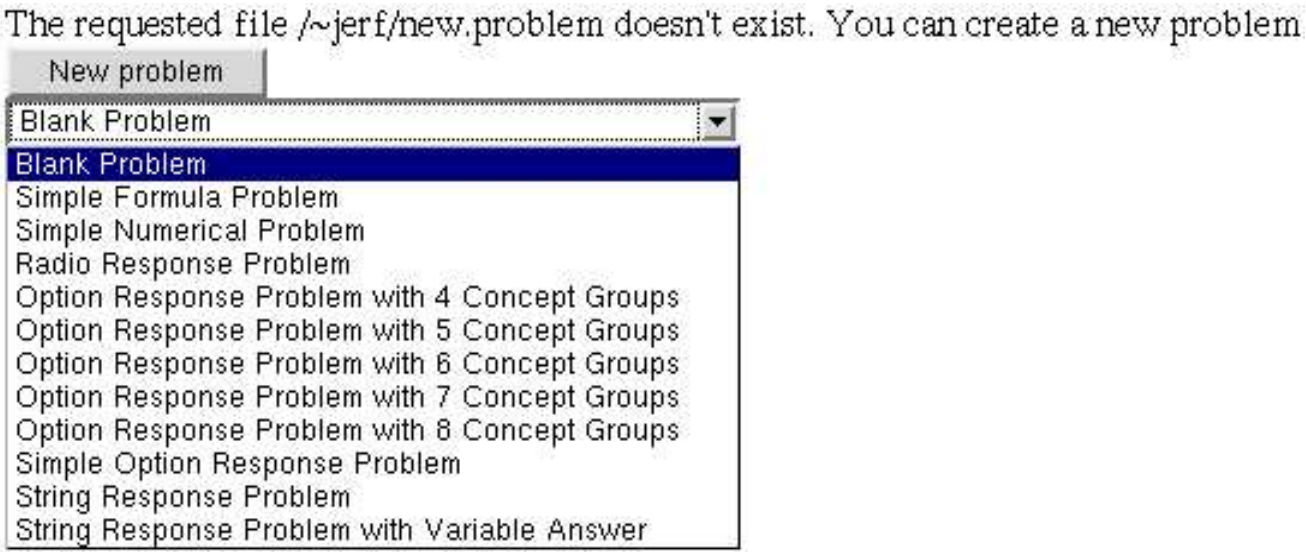


Figure 5: Creating A New Problem Resource

2.7 Creating Radio Response Problems

To create a **Radio Response** problem, create a new resource as described in section 2.2. 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. In the drop-down option box, select **Radio Response Problem**, and click the **New Problem** button.

2. Click the **Edit** button above the sample problem to enter edit mode. You should see an editing screen.

3. 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?"

4. 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.

5. 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.

6. 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.

7. Repeat the previous step until you've filled in all of the other incorrect answers you wish to offer the students.

8. Once you've filled in all of the incorrect answers, change the **Correct Option**s on the other foils to **Unused**.

Figure 6: Radio Response Creation Form

9.

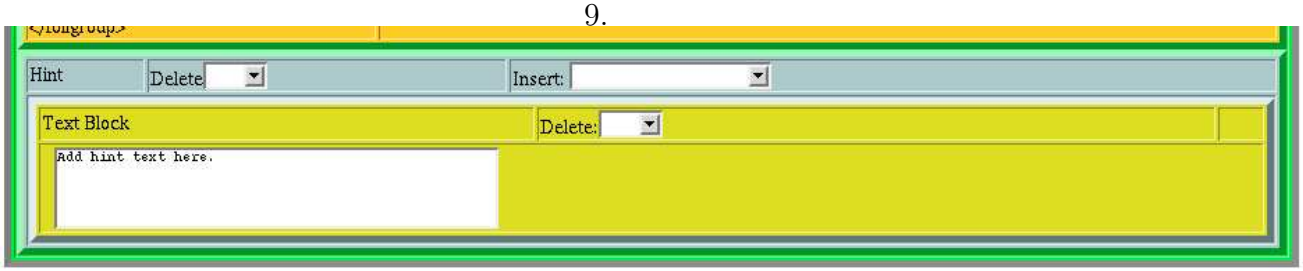


Figure 7: Hint Element

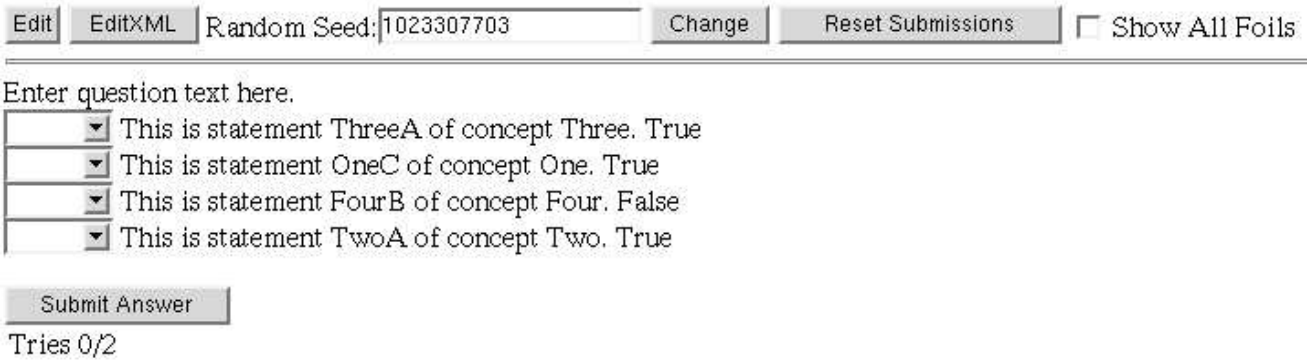


Figure 8: Option Response Problem

10. Scroll down to the Hint element. 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.

11. 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.

2.7.1 Randomization

LON-CAPA will randomize the choices presented to each student and the order they are presented in. If you wish to present each student the same choices, make sure the **Maximum Number of Shown Foils** box contains the number of incorrect answers, which will force them to all be displayed.

2.8 Option Response Problems

2.8.1 Option Response Problems with Concept Groups

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

1. The Concept Groups

2. The options for the students to select, by default "True" and "False"

3. 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.

2.8.2 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.

2.8.3 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.

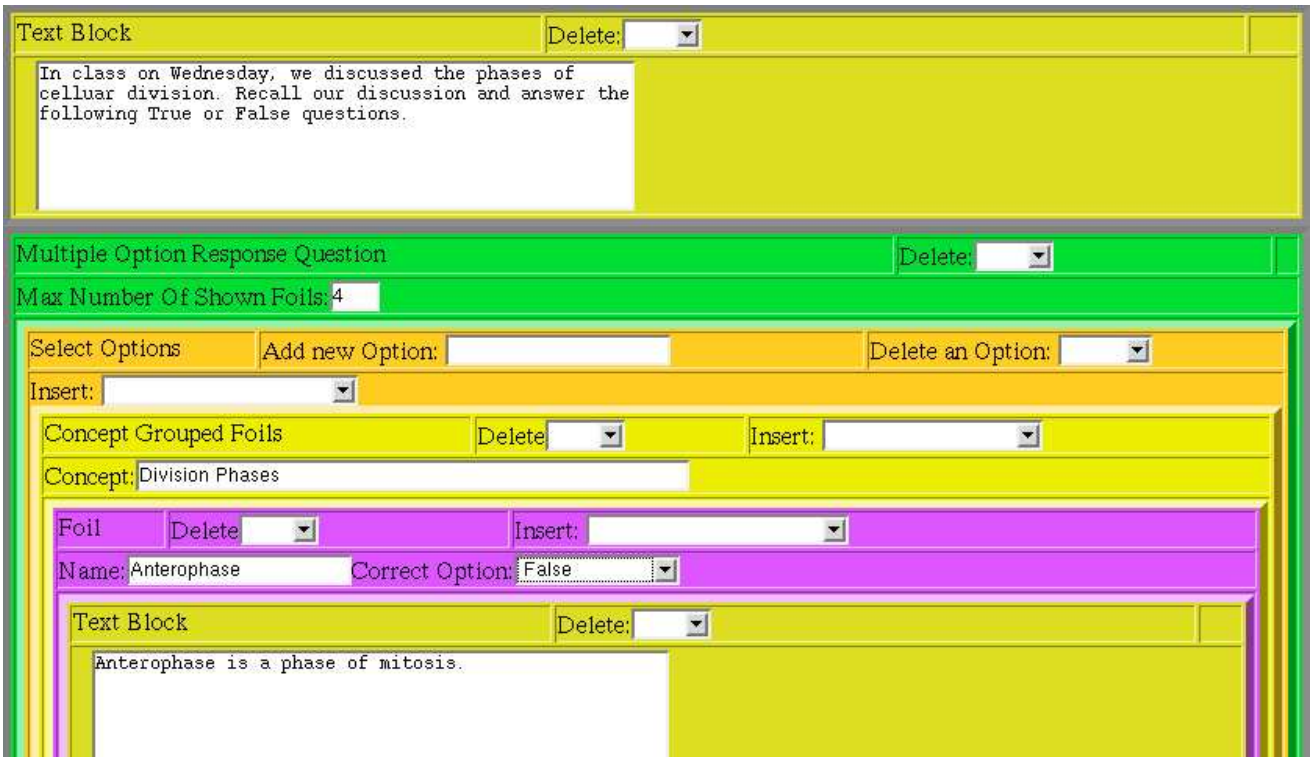


Figure 9: Option Response Editor

2.8.4 Creating Option Problems

To create an Option Response problem, create a new resource as described in section 2.2. This is a "problem" resource so the URL must end in ".problem". You should see a screen as in figure "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.

2.8.5 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.

2.9 Creating a String Response Problem

To create a **String Response** problem, create a new resource (described in 2.2). This is a "problem" resource so the URL must end in ".problem".

1. In the drop-down option box as seen in 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 "String Response Editor" figure.

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.

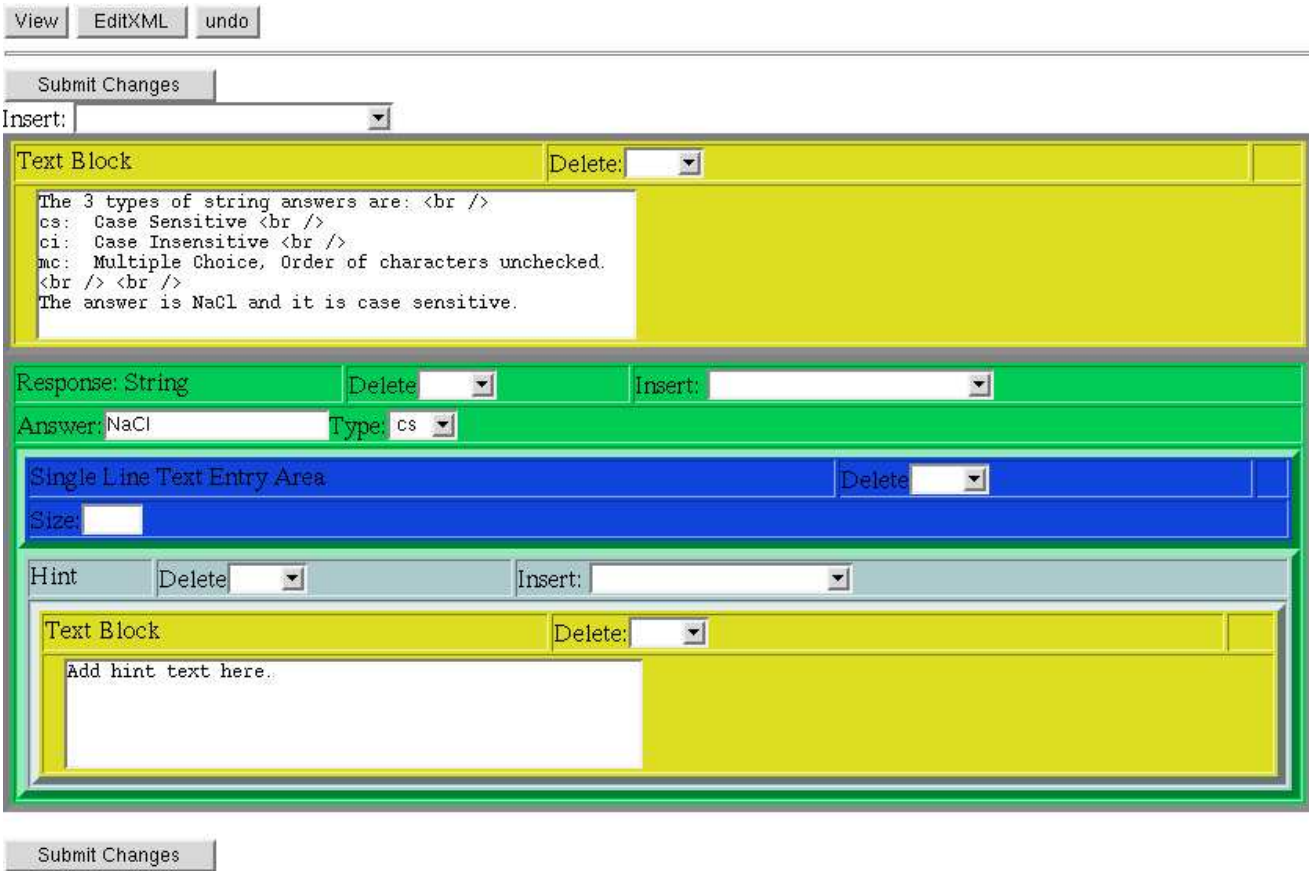


Figure 10: String Response Editor

(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.

2.10 Creating 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 $x^2 + 21 - 10$.

Creating Numerical Response and Formula Response problems starts the same as the other problem types, but because of the power of Numerical Response and Formula Response problems, they are covered in their own section after the end of the tutorial. For more information about these problem types, please see section 5 for Numerical Response problems and section 2.6.5 for Formula Response problems.

2.11 Dynamically Generated Plots

The **gnuplot** tag allows an author to design a plot which is created when it is viewed. This is intended for use in homework problems where each student needs to see a distinct plot. It can be used in conjunction with a **script** tag to generate random plots.

The following parameters may be set:

- `brief description of the plot` This text is used as the `alt` parameter of the `img` tag used to embed the plot.

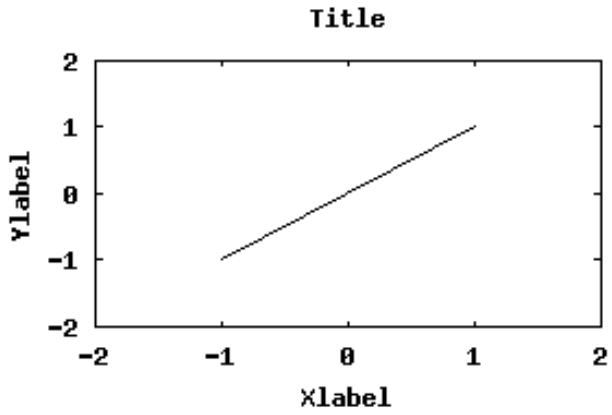
- `background color of image (xffffff)` See the section on color selection 2.13 for help on specifying colors.

- `foreground color of image (x000000)` See the section on color selection 2.13 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** 2.12 tag is used to specify the **curve** 2.12, 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 **gnuplot** tag allows the use of the the following tags:

- **curve** 2.12
- **key** 2.11
- **label** 2.11
- **axes** 2.11
- **tics** 2.11
- **title**, **xlabel**, and **ylabel** 2.11

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** 2.11 tag. The figure below shows the locations of the various labels.



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** 2.11 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 2.13.

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

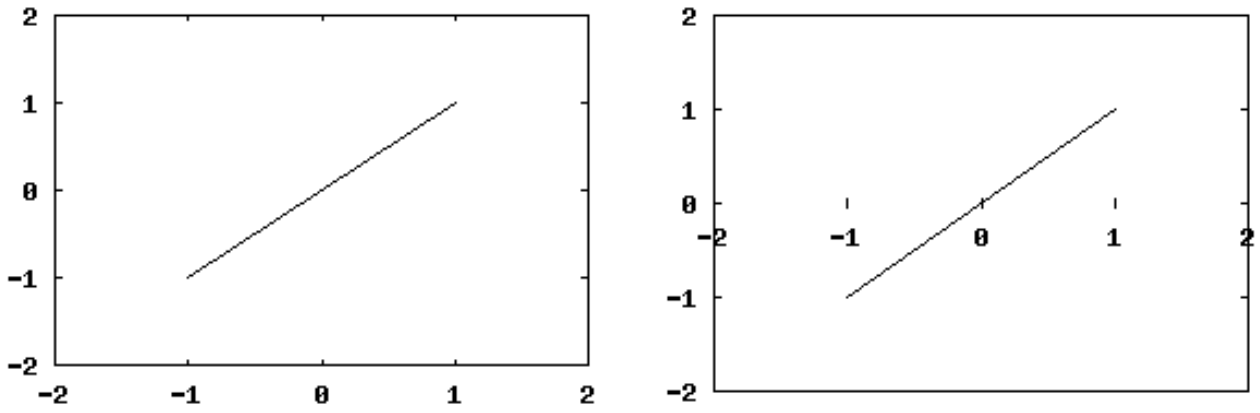
See also Plot Ticks 2.11 and the general Gnuplot help **??**.

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

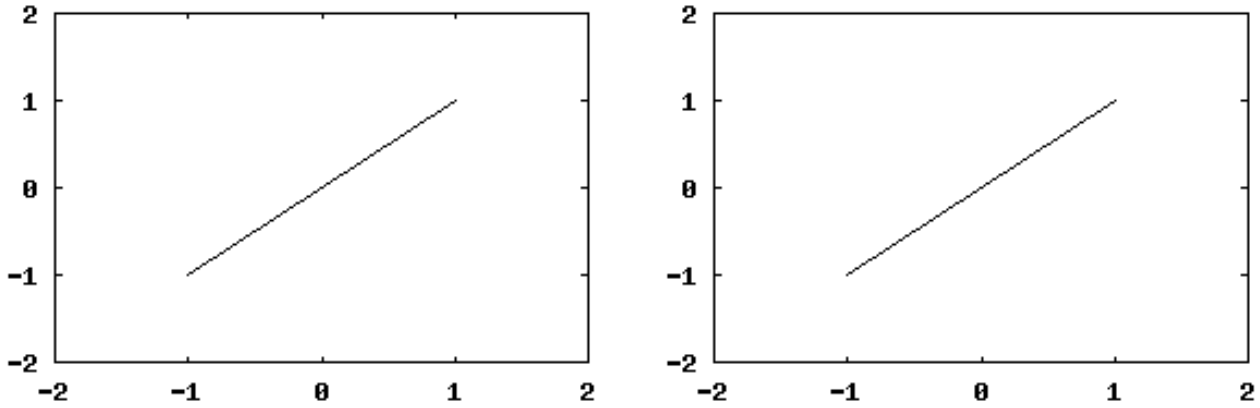
The **xtics** and **ytics** tags have identical structure and the description presented here applies to both.

The tics tags allow specification of the following parameters:

- `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`

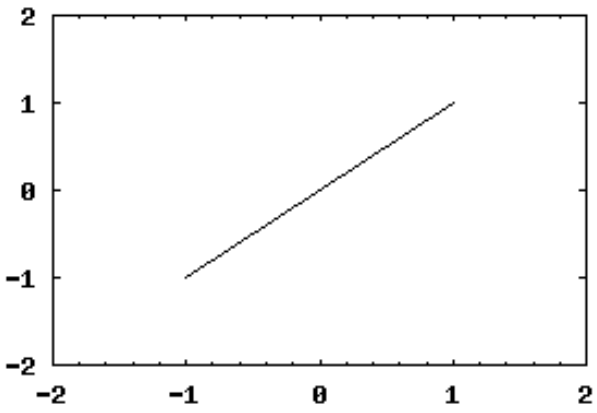
  The span, in graph coordinates, between each major tic mark.

- `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.



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** 2.12 which has a name.

The key is the color of the foreground of the plot, specified in the **gnuplot** 2.11 tag.

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** 2.11 tag.

- `justification of the label text on the plot` "left", "right", or "center".
- `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.

2.12 Specifying Curves to Plot

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

The following parameters may be set:

- color of curve

  The color of the curve on the plot. See **Selecting Colors** 2.13.

- name of curve to appear in key

  If a key is present, the name of the curve will appear with a sample of its line type.

- line style

  See the section on line styles 2.12 for more information about the available line styles and their data requirements.

- point type

  This parameter may not apply to all linestyles.

- point size

  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.

There are two ways of entering the information to be plotted, which are accessed using the subtags of **curve**, **data** 2.12 and **function** 2.12.

The **data** tag is used to specify the values plotted in the **gnuplot** 2.11 tag. The **data** tag is only used in the **Curve** 2.12 tag.

The data must be either a perl array, `@X`, or a comma seperated list, such as "0.5,0.9,1.5, 2.4" (without quotes). 'NaN' is a valid value.

The function and number **data** tags required varies based on the line style 2.12 chosen for the curve. 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.

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*)

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.

2.13 Color Selection

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

- Background color is an attribute of the gnuplot tag 2.11. This controls the color of the plot image. The default is white (xffffff).

- Forground color is an attribute of the gnuplot tag 2.11. This controls the color of the border.

- Gridline color is an attribute of the axis tag 2.11.

- Curve color is an attribute of the curve tag 2.12. This is the color of the curve function or data points. Different curves can be given different colors.

2.14 General Problem Editing

The following capabilities are available in all problem types:

2.14.1 Adding Picture

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

```
<img src="/res/DOMAIN/AUTHOR/directory-of-picture/picture.jpg" />
```

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 then 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`.

3 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.

3.1 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. 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, which 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.

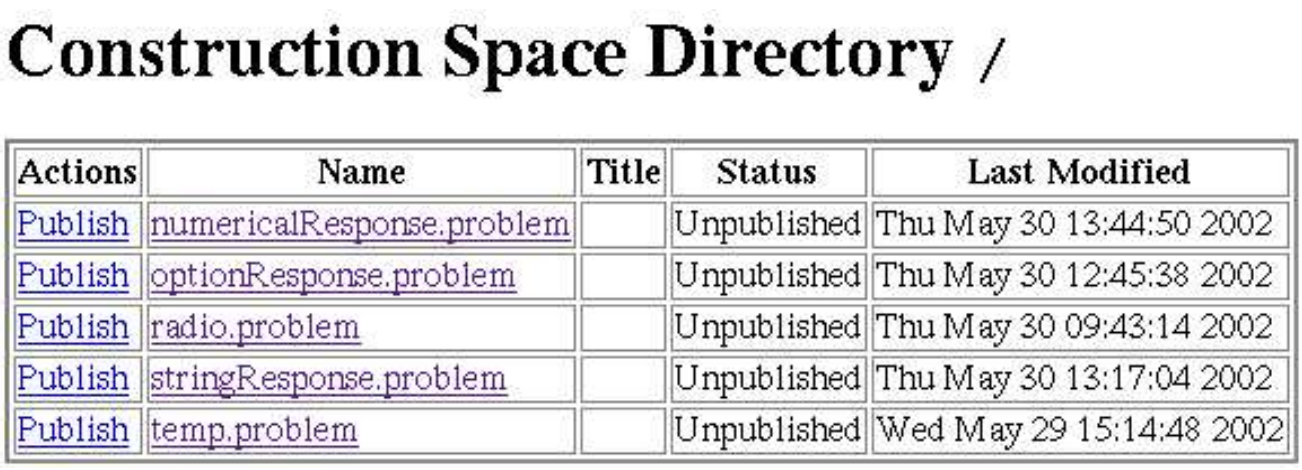


Figure 11: Construction Space for Publishing

One example of metadata is the ¡title¿ 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.

3.2 Publishing A Resource

To publish a resource, log in and choose your Author role. Then click **CSTR** to go to your construction space. You should see something like the "Construction Space for Publishing". Click on the **Publish** button 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 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** is the default. The content can be used for any course within the network, regardless of the domain. Instructors can find your content and use it in their courses. Once an instructor selected a resource, the students in the course have access.

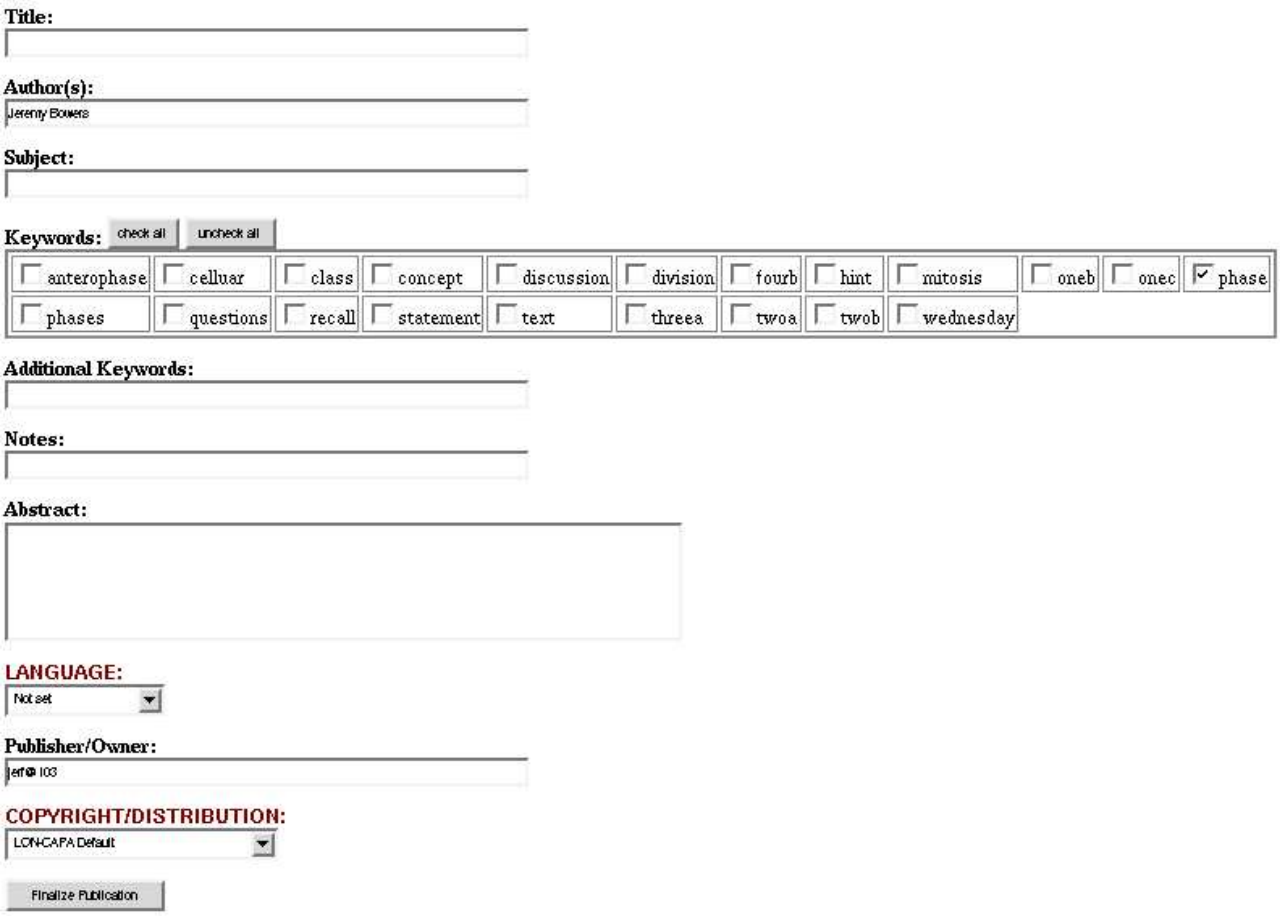


Figure 12: Publishing Metadata Screen

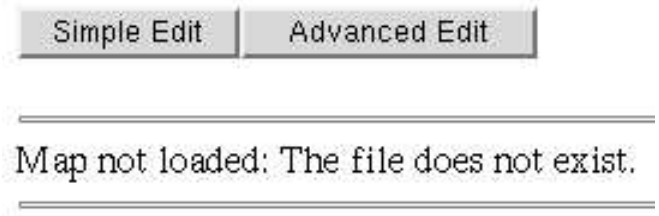


Figure 13: Map Editor Selection

- **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** means that it can't be used for any course.
- **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.

4 Creating A Course: Maps and Sequences

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

4.1 Creating 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 section 2.2. 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 13. You can use either the advanced editor or the simplified editor.

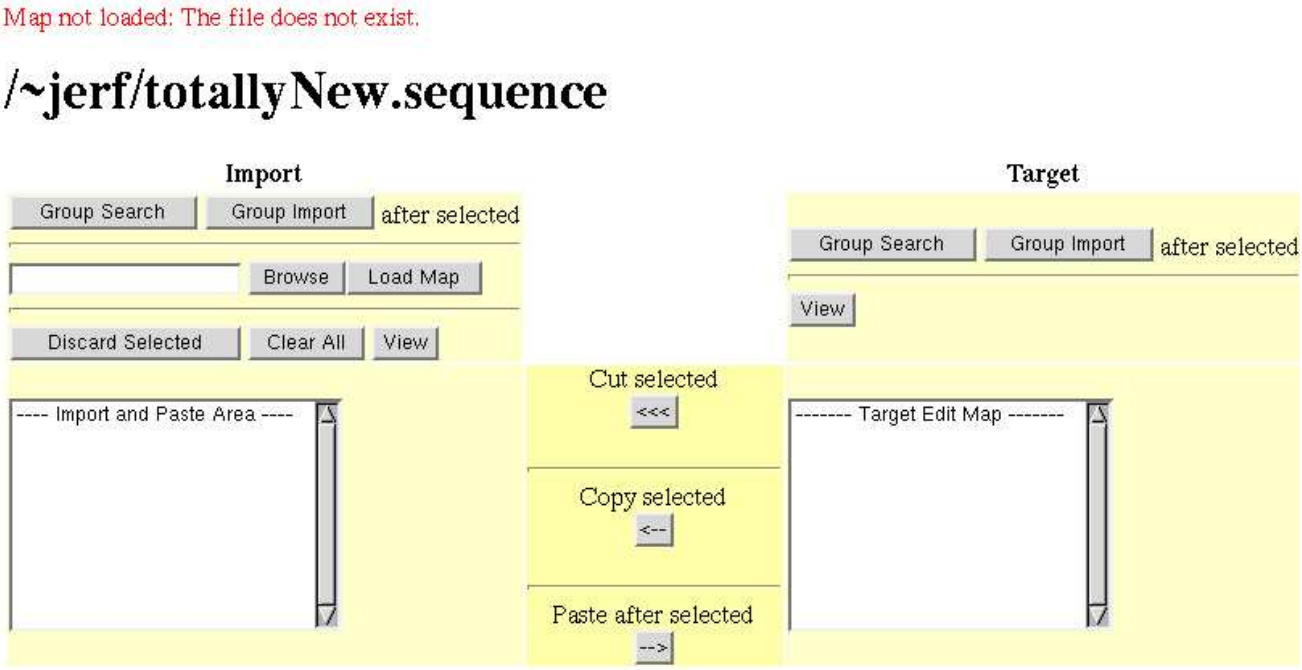


Figure 14: Simple Map Editor

4.2 Creating 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 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.

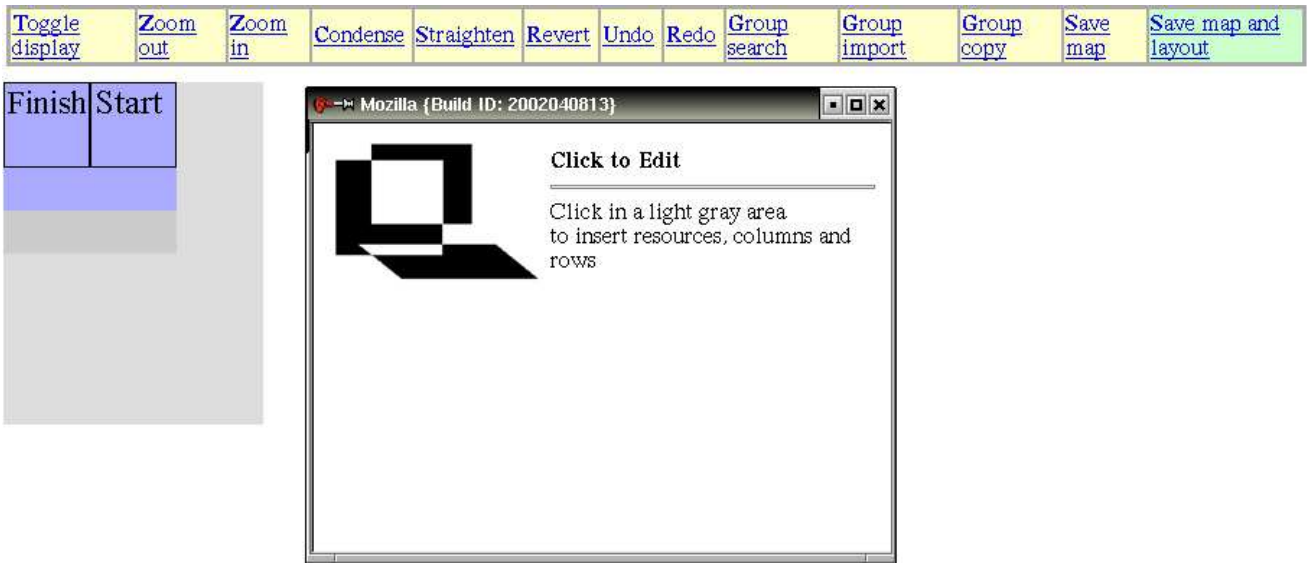


Figure 15: Initial Map Editor

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

4.3 Creating a Simple .sequence With The Advanced Editor

After creating a new .sequence resource and getting the editor selection prompt (13), 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.

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

1.

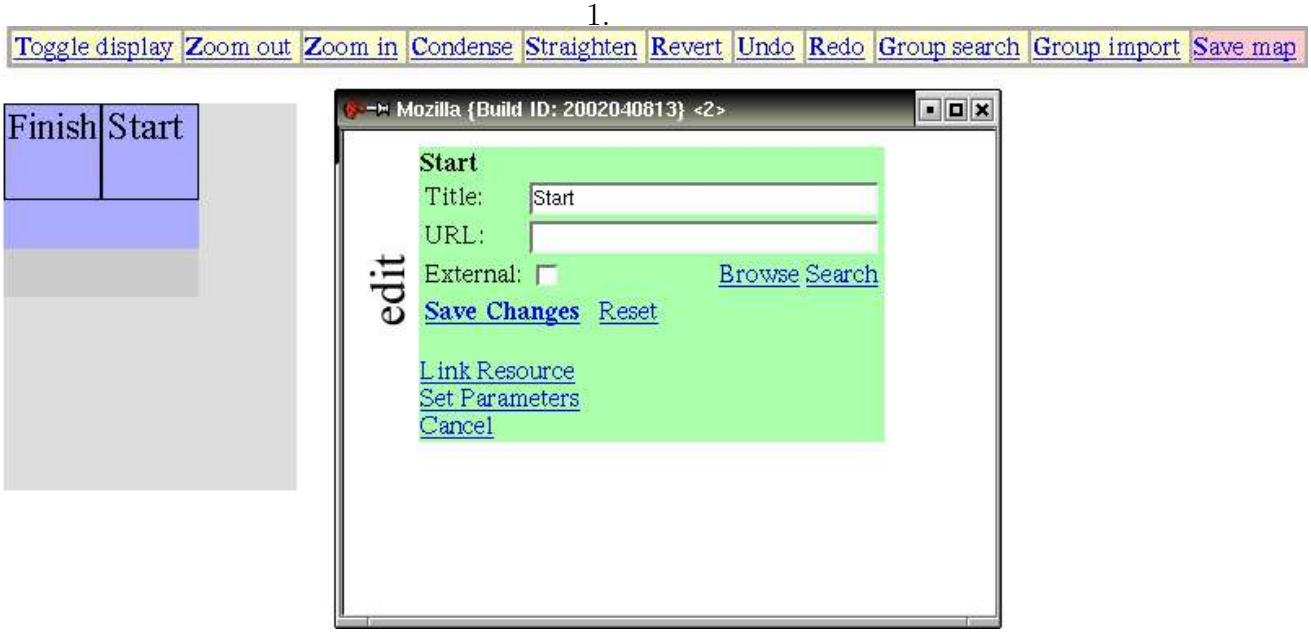


Figure 16: After clicking **Start** in the Map Constructor

Figure 17: Straightened Map

2.

Figure 18: Inserting a Resource

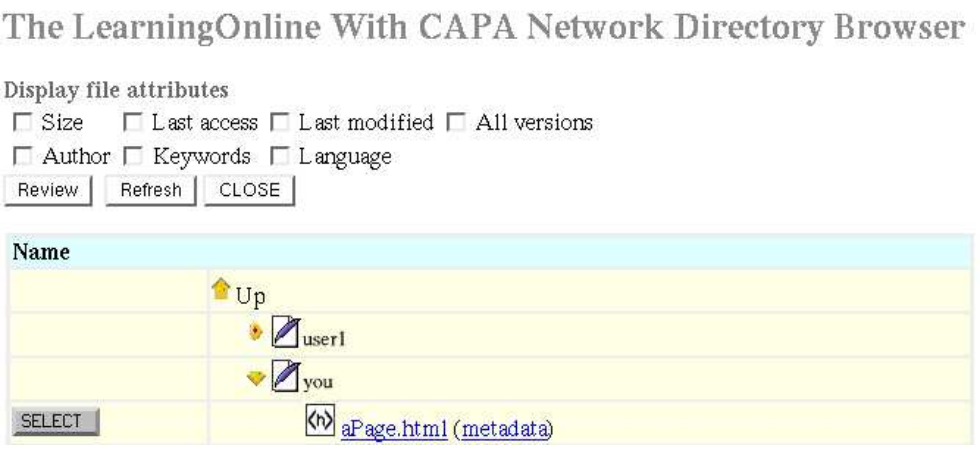


Figure 19: Network Directory Browser

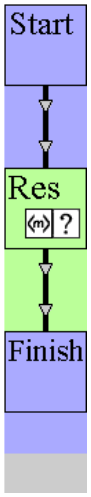


Figure 20: Resource Chosen

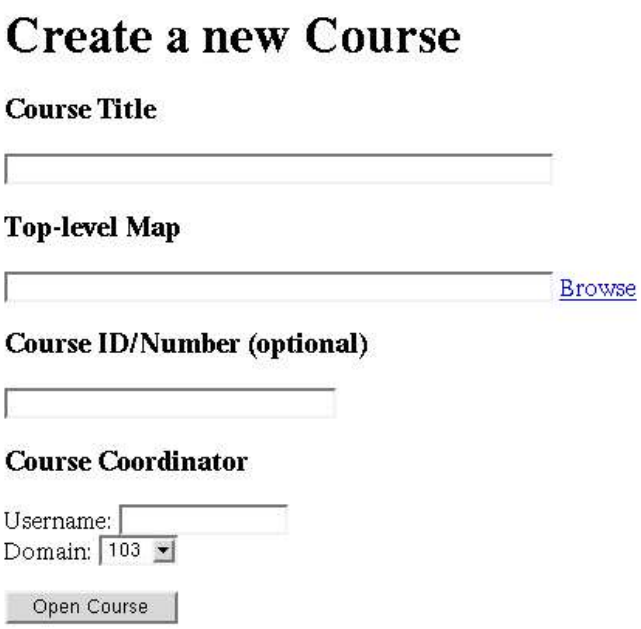


Figure 21: Creating a New Course

be set and the icons for the resource will appear in the **Res** box, as shown in figure 20. 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.

4.4 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.

4.5 Creating a Course: 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. Only Domain Coordinators can make courses and set their Top-level maps, so work with your Domain Coordinator if you need to view your maps.

5 Numerical Response And Formula Response Questions

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 section 2.2, ending your resource name with ".problem", and create a new **Simple Numerical Response** problem.

5.1 The Parts of a Numerical Response Problem

A Numerical Response problem has seven major parts by default:

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. This can also use parameters from the **Script** block, allowing the answer to be computed dynamically.

4. A **tolerance** parameter determines how closely the system will require the student's answer to be in order to count it correct.

   For technical reasons, it is almost never a good idea to set this parameter to 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 a small integer, you *need* to allow a tolerance factor, or the students will find it nearly impossible to exactly match the computers idea of the answer. You may find the default too large for some problems.

   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.

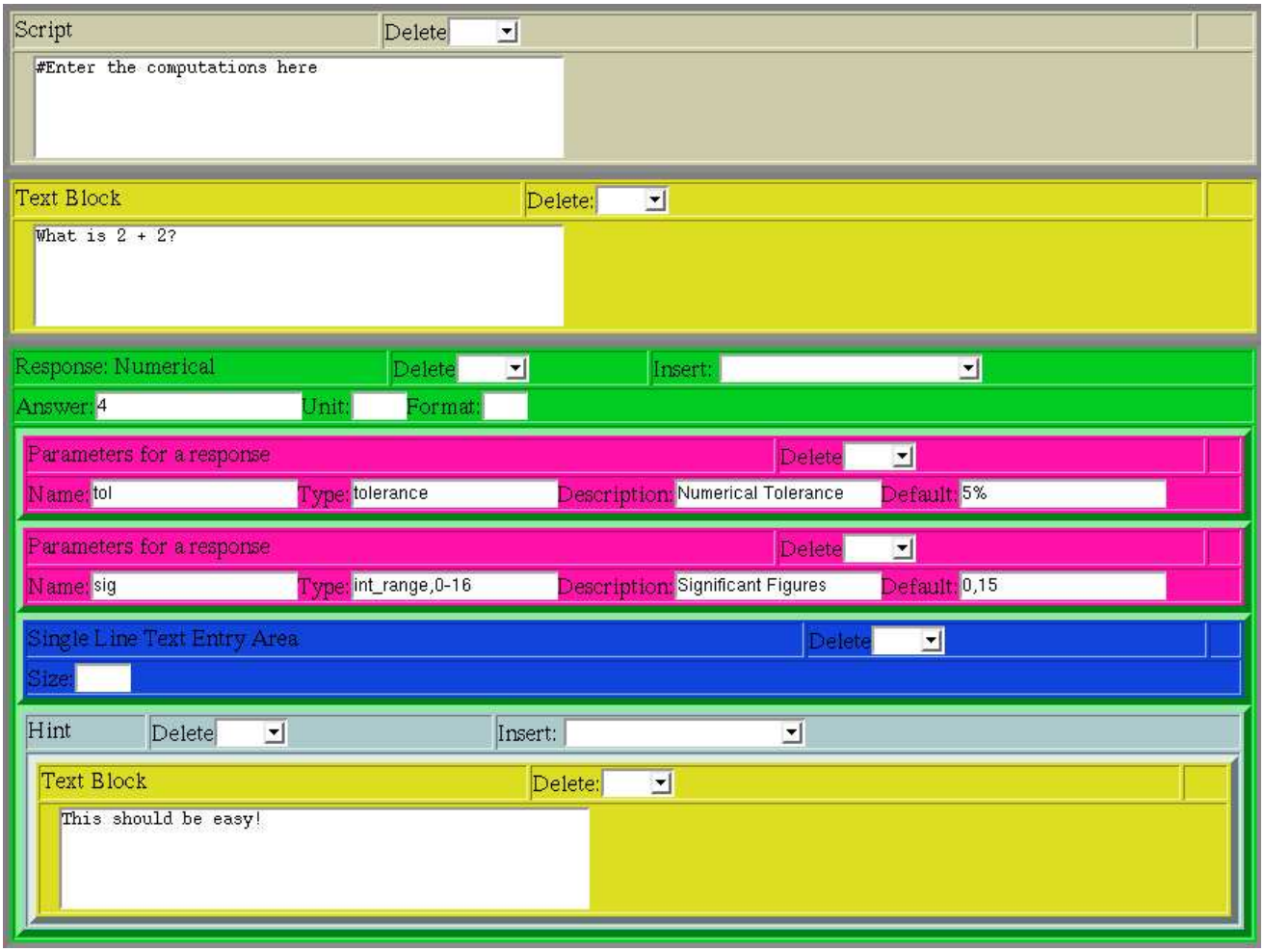


Figure 22: Numerical Response editor

(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)

5. A **significant figures** specification tells the system how many significant figures there are in the problem, as either a single number or a range of acceptable values, expressed as **min,max**. The system will check to make sure that the student's answer contains this many significant digits, useful in many scientific calculations. For example, if the problem has three significant digits, the significant digit specification is "3", and the answer is "1.3", 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 "3,4" means both "1.30" and "1.300" are acceptable.

6. The **Single Line Text Entry** area, as in other problem types, allowyou 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.

5.2 Simple Numerical Response Answer

Along with showing the Numerical Response editor, figure 22 also shows the parameters for one of the simplest possible types of numerical response. 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.

5.3 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 variables[1]. 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.

5.3.1 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.'';
```

[1]The author favors `capsOnNewWords`. Some people use `underscore_to_separate_words`. Many use uppercase letters to specify constants like `PI` or `GOLDEN_MEAN`. Some people always `StartWithCapatalization`. 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.

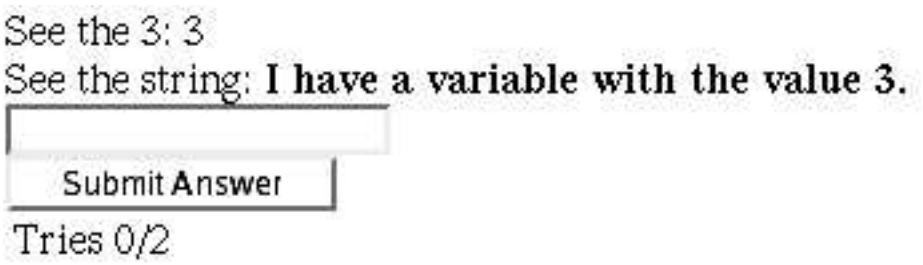


Figure 23: Result of Variables in the Text Block

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.

5.3.2 Variables in the Text Block

Once you've defined variables in the **Script**, you can use 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 23. Note how the "$variable" was turned into a 3, and the "$stringVar" was turned into "I have a variable with the value 3."

5.3.3 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%.

5.4 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 at **??**.

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

In the **Script** block, function names start with **&**. Some example function calls are shown in figure 24. 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.

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

Figure 24: Some Function Calls

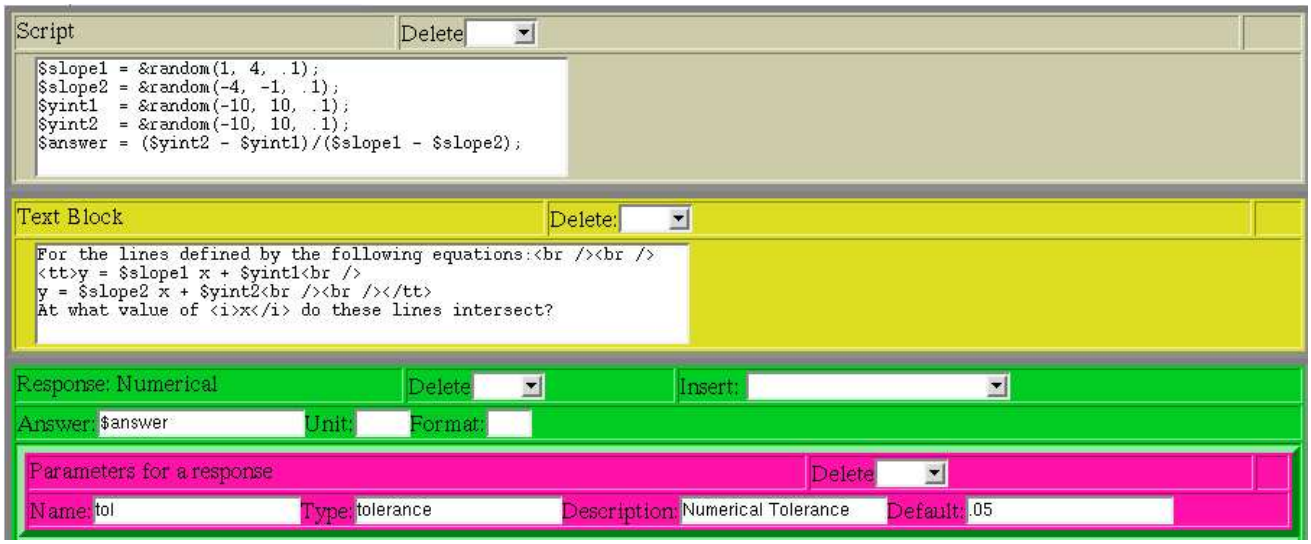


Figure 25: Slope Problem Parameters

5.4.1 Numerical Response Randomization

If you're doing this as a tutorial, try a few random seeds to see what happens.

5.5 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.

Try filling out your problem with the parameters shown in the "Slope Problem Parameters" figure.

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.

5.6 Units, Format

Numerical Response problems can require units. In the problem editing form, place the desired unit in the **Unit** field. For information about what units the system accepts, see **??**.

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.

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 ".333333333" 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.

5.7 For More Information

The full power of Perl is well outside the scope of this document. Looking in the function list at 7.3 can give you some ideas. O'Reilly has some good Perl books. The Perl 5 Pocket Reference will contain more than what you need to know to use LON-CAPA, available at http://www.oreilly.com/catalog/perlpr3/ .

If you have any problems, consult http://help.loncapa.org/fom/cache/5.html . If you don't find the answer to your problem, please help us expand the FAQ by submitting a new pending question.

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.

5.8 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.

5.8.1 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 $\sin 2x = 2 \sin x \cos x$? Symbolically proving it one way or another is impossible in general. Therefore, LON-CAPA uses a sampling system. 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 $2x^2 + 3y^3$, which has two variables, one could specify:

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

5.8.2 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.

5.8.3 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 .

6 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.

6.1 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 Construction Space.

6.1.1 numericalresponse

stringresponse implements a string answer. An internal **textline** tag (see **6.5**) 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**: **c**ase **s**ensitive, order important.
  - **ci**: **c**ase **i**nsensitive, order important.
  - **mc**: case insensitive, order unimportant. The mnemonic for this option is "**m**ultiple **c**hoice", 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.

6.1.2 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>`. 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.

6.1.3 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 **??**, with this additional addition:

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

6.1.4 radiobuttonresponse

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

6.1.5 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.

6.1.6 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**: string or scalar variable that can encode something that should encode the correct answer. In some cases, this may be nothing.

- **form**: hash variable name that will be submitted to the remote site as a HTTP form.

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

- **loncapagrade**: takes no attributes, but must surround the response.
- **awardetail**: 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.awarddetail**.
- **message**: optional message to have shown to the student.

Example:

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

6.1.7 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.

6.2 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.

6.3 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**.

6.4 Hint Tags

All of these tags must appear inside a response tag:

- **hintgroup**: tag that surrounds all of a hint.
- **hintpart**: required. 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 a two element list, first the type (**foil** or **concept**) and then either the foil's name or the concept's string.

6.5 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.

6.6 Output Tags

This group of tags generates useful output.

- **algebra**: Typesets algebraic expressions

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

- **chem**: Typesets chemical equation

```
<chem>02 + 2H2 -> 2H20</chem>
```

- **num**: Typesets a number

```
<num format=''2E''>31454678</num>
```

- **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 inbetween 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.

- **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 stylesheets.

- **window**: The text inbetween is put in a pop-up javascript window.

- **m**: The inside text is LaTeX, and is converted to HTML (or MathML) on the fly. If the attribute **eval** is set to "**on**" the intervening text will have a perl variable expansion done to it before being converted. The default is to convert to the display mechanism that the user has selected. This can be overriden by setting the attribute **display** to one of "**tth**" or "**jsMath**" or "**mimetex**"which will force a specfic display mechanism.

- **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>
```

  - **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.

6.7 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.

6.8 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** restricts the number of tags inside that are actually parsed to no more than **show**.
- **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.

6.9 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.

7 <script> Tag

7.1 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)
- asin(x), acos(x), atan(x), atan2(y,x)
- log(x), log10(x)
- exp(), pow(x,y), sqrt(x)
- abs(x), sgn(x)
- erf(x), erfc(x)
- ceil(x), floor(x)
- min(...), max(...)
- factorial(n)
- N%M (modulo function)
- sinh(x), cosh(x), tanh(x)
- asinh(x), acosh(x), atanh(x)
- roundto(x,n)
- web("a","b","c") or web(a,b,c)

- html("a") or html(a)
- j0(x), j1(x), jn(n,x), jv(y,x)
- y0(x), y1(x), yn(n,x), yv(y,x)
- random
- choose
- tex("a","b") or tex(a,b)
- var_in_tex(a)
- to_string(x), to_string(x,y)
- class(), section()
- name(), student_number()
- check_status(partid)
- open_date(), due_date(), answer_date()
- sub_string()
- array_moments(array)
- format(x,y),prettyprint(x,y,target),dollarformat(x,target)
- map(...)
- caparesponse_check
- caparesponse_check_list

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
```

7.2 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

7.3 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

LON-CAPA Function	Description
&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 *erfx(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 and integer rounded down. If x is an integer than 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 $x as nn where nn is nF or nE or nS and n is an integer.
&prettyprint($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.

LON-CAPA Function	Description
&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.
&roundto($x,$n)	Rounds a real number to n decimal points. $x and $n can be pure numbers
&web("a","b","c") or &web($a,$b,$c)	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
&html("a") or &html($a)	Output only if the output mode chosen is in html format
&j0($x), &j1($x), &jn($m,$x), &jv($y,$x)	Bessel functions of the first kind with orders 0, 1 and m respectively. For jn(m,x), m must be an integer whereas for jv(y,x), y is real. $x can be a pure number. $m must be an integer and can be a pure integer number. $y can be a pure real number
&y0($x), &y1($x), &yn($m,$x), &yv($y,$x)	Bessel functions of the second kind with orders 0, 1 and m respectively. For yn(m,x), m must be an integer whereas for yv(y,x), y is real. $x can be a pure number. $m must be an integer and can be a pure integer number. $y can be a pure real number
&random($l,$u,$d)	Returns a uniformly distributed random number between the lower bound, l and upper bound, u in steps of d. $l, $u and $d can be pure numbers
&choose($i,...)	Choose the ith 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. $i can be a pure integer

LON-CAPA Function	Description
Option 1 - &map($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &map($seed,\@mappedArray,[$a,$b,$c,$d]) Option 3 - @mappedArray = &map($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &map($seed,\@a) Option 5 - @Z = &map($seed,\@a) where $a='A' $b='B' $c='B' $d='B' $w, $x, $y, and $z are variables	Assigns to the variables $w, $x, $y and $z the values of the $a, $b, $c and $c (A, B, C and D). The precise value for $w .. depends on the seed. (Option 1 of calling map). In option 2, the values of $a, $b .. are mapped into the array, @mappedArray. The two options illustrate the different grouping. Options 3 and 4 give a consistent way (with other functions) of mapping the items. For each option, the group can be passed as an array, for example, [$a,$b,$c,$d] => \@a. And Option 5 is the same as option 4 the array of results are saved into a signle array rather than an array opf scalar variables.
Option 1 - &rmap($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &rmap($seed,\@rmappedArray,[$a,$b,$c,$d]) Option 3 - @rmapped_array = &rmap($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &rmap($seed,\@a) Option 5 - @Z = &map($seed,\@a) where $a='A' $b='B' $c='B' $d='B' $w, $x, $y, and $z are variables	The rmap functions does the reverse action of map if the same seed is used in calling map and rmap.
$a=&xmlparse($string)	You probably should use the tag <parse> 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.
&tex($a,$b), &tex("a","b")	Returns a if the output mode is in tex otherwise returns b
&var_in_tex($a)	Equivalent to tex("a"," ")

LON-CAPA Function	Description
&to_string($x), &to_string($x,$y)	If x is an integer, returns a string. If x is real than the output is a string with format given by y. For example, if x = 12.3456, &to_string(x,".3F") = 12.345 and &to_string(x,".3E") = 1.234E+01.
&class(), &section()	Returns null string, class descriptive name, section number, set number and null string.
&name(), &student_number()	Return the full name in the following format: lastname, firstname initial. Student_number returns the student 9-alphanumeric string. If undefined, the functions return null.
&check_status($partid)	Returns a number identifing the current status of a part. Ture values mean that a part is "done" (either unanswerable becuase of tries exhuastion, or correct) or a false value if a part can still be attempted. If $part is unspecfied, it will check either the current <part>'s status or if outside of a <part>, check the status of previous <part>. The full set of return codes are: 'undef' means it is unattempted, 0 means it is attmpted and wrong but still has tries, 1 means it is marked correct, 2 means they have exceed maximum number of tries, 3 means it after the answer date
&open_date(), &due_date(), &answer_date()	Problem open date, due date and answer date. The time is also included in 24-hr format.
Not implemented	Get and set the random seed.
&sub_string($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.
@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

LON-CAPA Function	Description
@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 $item_cnt deviates of beta distribution. The density of beta is: Xˆ($aa-1) *(1-X)ˆ($bb-1) /B($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**$r)/gamma($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). $nonc is the noncentrality parameter.
@return_array=&random_multivariate_normal ($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 obervation is equal to $item_cnt.

LON-CAPA Function	Description
@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 probabilty 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 probabilty of an event in each trial is $p.

7.4 Table: CAPA vs. LON-CAPA function differences

CAPA Functions	LON-CAPA	Differences (if any)
sin(x), cos(x), tan(x)	&sin($x), &cos($x), &tan($x)	
asin(x), acos(x), atan(x), atan2(y,x)	&asin($x), &acos($x), &atan($x), &atan2($y,$x)	
log(x), log10(x)	&log($x), &log10($x)	
exp(x), pow(x,y), sqrt(x)	&exp($x), &pow($x,$y), &sqrt($x)	
abs(x), sgn(x)	&abs($x), &sgn($x)	
erf(x), erfc(x)	&erf($x), &erfc($x)	
ceil(x), floor(x)	&ceil($x), &floor($x)	
min(...), max(...)	&min(...), &max(...)	
factorial(n)	&factorial($n)	
N%M	$N%$M	
sinh(x), cosh(x), tanh(x)	&sinh($x), &cosh($x), &tanh($x)	
asinh(x), acosh(x), atanh(x)	&asinh($x), &acosh($x), &atanh($x)	
/DIS($x,"nn")	&format($x,'nn')	The difference is obvious.
Not in CAPA	&prettyprint($x,'nn','optional target')	
Not in CAPA	&dollarformat($x,'optional target')	
roundto(x,n)	&roundto($x,$n)	
web("a","b","c") or web(a,b,c)	&web("a","b","c") or &web($a,$b,$c)	
html("a") or html(a)	&html("a") or &html($a)	

CAPA Functions	**LON-CAPA**	**Differences (if any)**
jn(m,x)	&j0($x), &j1($x), &jn($m,$x), &jv($y,$x)	In CAPA, j0, j1 and jn are contained in one function, jn(m,x) where m takes the value of 0, 1 or 2. jv(y,x) is new to LON-CAPA.
yn(m,x)	&y0($x), &y1($x), &yn($m,$x), &yv($y,$x)	In CAPA, y0, y1 and yn are contained in one function, yn(m,x) where m takes the value of 0, 1 or 2. yv(y,x) is new to LON-CAPA.
random(l,u,d)	&random($l,$u,$d)	In CAPA, all the 3 arguments must be of the same type. However, now you can mix the type
choose(i,...)	&choose($i,...)	
/MAP(seed;w,x,y,z;a,b,c,d)	Option 1 - &map($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &map($seed,\@mappedArray,[$a,$b,$c,$d]) Option 3 - @mappedArray = &map($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &map($seed,\@a) where $a='A' $b='B' $c='B' $d='B' $w, $x, $y, and $z are variables	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.

CAPA Functions	LON-CAPA	Differences (if any)
rmap(seed;a,b,c,d;w,x,y,z)	Option 1 - &rmap($seed,[\$w,\$x,\$y,\$z],[$a,$b,$c,$d]) or Option 2 - &rmap($seed,\@rmappedArray,[$a,$b,$c,$d]) Option 3 - @rmapped_array = &rmap($seed,[$a,$b,$c,$d]) Option 4 - ($w,$x,$y,$z) = &rmap($seed,\@a) where $a='A' $b='B' $c='B' $d='B' $w, $x, $y, and $z are variables	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 unamed 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	$a=&xmlparse($string)	New to LON-CAPA
tex(a,b), tex("a","b")	&tex($a,$b), &tex("a","b")	
var_in_tex(a)	&var_in_tex($a)	
to_string(x), to_string(x,y)	&to_string($x), &to_string($x,$y)	
capa_id(), class(), section(), set(), problem()	&class(), &section()	capa_id(), set() and problem() are no longer used. Currently, they return a null value.
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.

CAPA Functions	LON-CAPA	Differences (if any)
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: $av MUST be non-negative.	In CAPA the results are passed as the first argument whereas in LON-CAPA the results are set equal to the function.
random_poisson (return_array,item_cnt,seed,mu)	@return_array=&random_poisson ($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

CAPA Functions	LON-CAPA	Differences (if any)
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 $nonc must be non-negative.	New to LON-CAPA
NOT DOCUMENTED IN CAPA	@return_array=&random_multivariate_normal ($item_cnt,$seed,\@mean,\@covar) NOTE: @mean should be of length p array of real numbers. @covar should be a length p array of references to length p arrays of real numbers (i.e. a p by p matrix.	Note the backslash before the @mean and @covar arrays.
NOT IMPLEMENTED IN CAPA	@return_array=&random_multinomial ($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 $high.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_uniform_integer ($item_cnt,$seed,$low,$high) NOTE: $low and $high are both passed through int(). $low must be less than or equal to $high.	New to LON-CAPA
NOT IMPLEMENTED IN CAPA	@return_array=&random_binomial ($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

8 Appendix: Symbols in Tex

8.1 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 enties	Copy this column
α	α or α	<m>α</m>
β	β or β	<m>β</m>
γ	γ or γ	<m>γ</m>
Γ	Γ or Γ	<m>Γ</m>
δ	δ or δ	<m>δ</m>
Δ	Δ or Δ	<m>Δ</m>
ϵ	ε or ε	<m>ϵ</m>
ε		<m>ε</m>
ζ	ζ or ζ	<m>ζ</m>
η	η or η	<m>η</m>
θ	θ or θ	<m>θ</m>
ϑ	ϑ or ϑ	<m>ϑ</m>
Θ	Θ or Θ	<m>Θ</m>
ι	ι or ι	<m>ι</m>
κ	κ or κ	<m>κ</m>
λ	λ or λ	<m>λ</m>
Λ	Λ or Λ	<m>Λ</m>
μ	μ or μ	<m>μ</m>
ν	ν or ν	<m>ν</m>
ξ	ξ or ξ	<m>ξ</m>
Ξ	Ξ or Ξ	<m>Ξ</m>
π	π or π	<m>π</m>
ϖ	ϖ or ϖ	<m>ϖ</m>
Π	Π or Π	<m>Π</m>
σ	σ or σ	<m>σ</m>
ς		<m>ς</m>
Σ	Σ or Σ	<m>Σ</m>
τ	τ or τ	<m>τ</m>
υ	υ or υ	<m>υ</m>
Υ	Υ or Υ	<m>Υ</m>
ϕ	φ or φ	<m>ϕ</m>
φ		<m>φ</m>
Φ	Φ or Φ	<m>Φ</m>
χ	χ or χ	<m>χ</m>
ψ	Ψ or ψ	<m>ψ</m>
Ψ	Ψ or Ψ	<m>Ψ</m>
ω	ω or ω	<m>ω</m>
Ω	Ω or Ω	<m>Ω</m>
ρ	ρ or ρ	<m>ρ</m>
ϱ		<m>ϱ</m>

8.2 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
$\pm$	± or ±	<m>$\pm$</m>
$\times$	× or ×	<m>$\times$</m>
$\div$	÷ or ÷	<m>$\div$</m>
$\cdot$	· or ·	<m>$\cdot$</m>
$\star$		<m>$\star$</m>
$\circ$	° or °	
$\bullet$	•	<m>$\bullet$</m>
$\dagger$		<m>$\dag$</m>
$\ddagger$		<m>$\ddag$</m>
†	&#134;	<m>$\dagger$</m>
‡	&#135;	<m>$\ddagger$</m>
©	© or ©	<m>$\copyright$</m>
$\leq$	≤ or ≤	<m>$\leq$</m>
$\geq$	≥ or ≥	<m>$\geq$</m>
$\neq$		<m>$\neq$</m>
$\ll$		<m>$\ll$</m>
$\gg$		<m>$\gg$</m>
$\simeq$		<m>$\simeq$</m>
$\perp$	⊥ or ⊥	<m>$\perp$</m>
$\parallel$		<m>$\parallel$</m>
$\leftarrow$	← or ←	<m>$\leftarrow$</m>
$\Leftarrow$	⇐ or ⇐	<m>$\Leftarrow$</m>
$\rightarrow$	→ or →	<m>$\rightarrow$</m>
$\Rightarrow$	⇒ or ⇒	<m>$\Rightarrow$</m>
$\uparrow$	↑ or ↑	<m>$\uparrow$</m>
$\Uparrow$	⇑ or ⇑	<m>$\Uparrow$</m>
$\leftrightarrow$	↔ or ↔	<m>$\leftrightarrow$</m>
$\Leftrightarrow$	⇔ or ⇔	<m>$\Leftrightarrow$</m>
$\surd$	√ or √	<m>$\surd$</m>
∂	∂ or ∂	<m>∂</m>
$\sum$	∑ or ∑	<m>$\sum$</m>
$\int$	∫ or ∫	<m>$\int$</m>
∞	∞ or ∞	<m>∞</m>