WeBWorK Homework Problems with Embedded Flash Applets

Day 1 Handout

1 Introduction

1.1 What is WeBWorK?

WeBWorK is a free Perl-based system for delivering individualized homework problems over the web. It was originally developed in 1995 by Profs. Arnold Pizer and Michael Gage at the University of Rochester Department of Mathematics, for use in mathematics instruction. It is now supported by a team of developers from several institutions and is used for a variety of subjects.

WeBWorK enhances the educational process in several ways. By providing students with immediate feedback as to the correctness of their answers, students are encouraged to make multiple attempts until they succeed. By individualizing problems, cheating is discouraged. By providing instructors with real-time statistics, lesson plans can be customized to better serve students.

The major way in which WeBWorK differs from other web-based homework systems is in the way problems are written. The PG ("problem generation") language allows the inclusion of both Perl and LaTeX code, allowing problem authors to take advantage of the syntactic efficiency of Perl and the typographical flexibility of LaTeX (which is pretty much necessary for rendering mathematics expressions).

The process of defining a problem is highly modular, with various pluggable display macros, answer evaluators, and graders. This allows for a high degree of freedom in defining how problems behave.

A library of ready to use problems is provided, with over 25,000 problems contributed to the National Problem Library (or NPL) at last count. A large collection of macro files which simplify writing new problems is also available. (Use the MathObjects macros provided by Davide Cervone wherever possible to simplify the writing of the problem and to make it easier to maintain.)

`http://webwork.maa.org/wiki/Introduction` `http://webwork.maa.org/` `http://webwork.maa.org/wiki/Main_Page`

1.2 A brief tour of the WeBWorK interface

- Classlist Editor – The Classlist editor is where the list of students enrolled in the class is maintained.
- Instructor Tools – This link provides access to a variety of instructor tools
- Homework Sets Editor
- Library Browser

- Student Progress
- File Manager

1.3 WeBWorK is built from Perl

We will not be discussing much about Perl in this minicourse though it is undoubtedly true that an understanding of Perl is sometimes helpful in programming in the PG language which we will use for writing WeBWorK homework problems. There are many excellent resources for learning more about Perl: `http://www.perl.org/`, `http://perldoc.perl.org/perlintro.html`. These include resources particular to WeBWorK and Perl: `http://webwork.maa.org/w/images/7/7a/Webwork-PREP-2011-Webconference1-Slides.pdf`, `http://webwork.maa.org/wiki/Basic_Perl_syntax#`.

About PG

- PG is built on Perl
- PG provides macros (prewritten, re-usable code)
- PG displays questions in two modes: HTML and PDF output
- MathObjects is an extension of PG written by Davide Cervone(Union College)

1.4 Overview

2 Basic setup

These instructions are designed to be step-by-step instructions on how to create problems in the WeBWorK MathFest Minicourse 2012. The outline of the steps is here, and each step is explained in greater detail below. In the first lesson we will log in to the WeBWorK MathFest Minicourse 2012 course, `https://hosted2.webwork.rochester.edu/webwork2/mathfest2012/`, use the Homework set editor to create a problem set for the homework problems you will create, generate 10 blank problems for your homework set, and create a file to edit using WeBWorK's editor.

Lesson 1 Logging in and Creating a Really Boring Problem Set

1. Log in to the WeBWorK MathFest Minicourse 2012 course. This should be straightforward. The URL is `https://hosted2.webwork.rochester.edu/webwork2/mathfest2012/`, and your username and password were provided to you.
2. On the left hand-side of your screen under Main Menu you will see "Instructor Tools". Under "Instructor Tools" select "Hmwk Sets Editor".

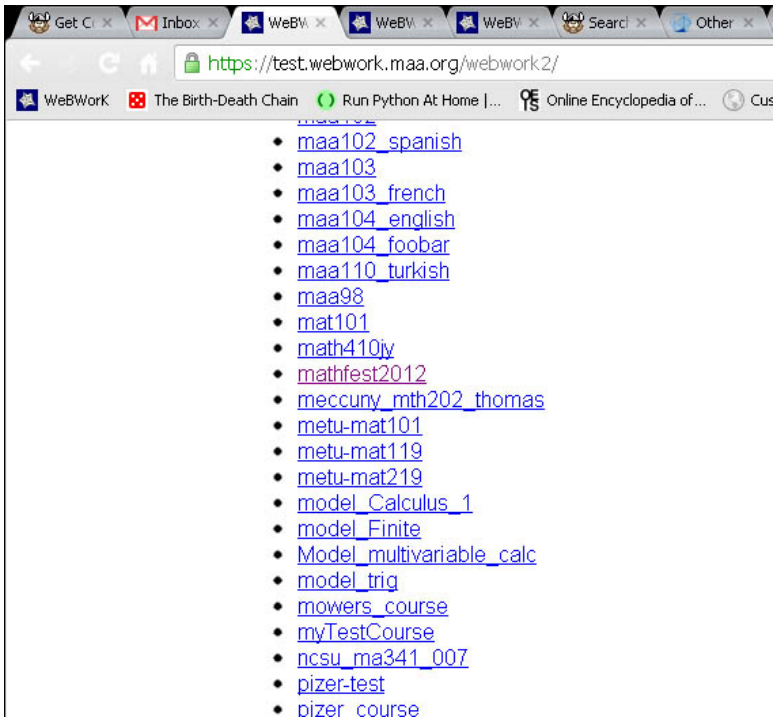


Figure 1: Select the mathfest2012 course from the list.

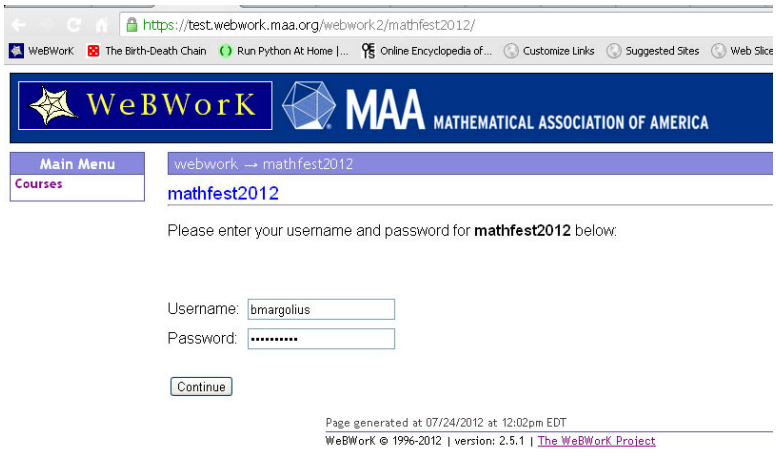


Figure 2: Enter your login information.

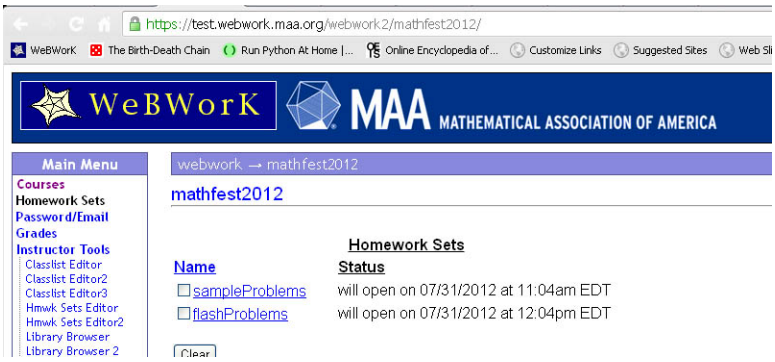
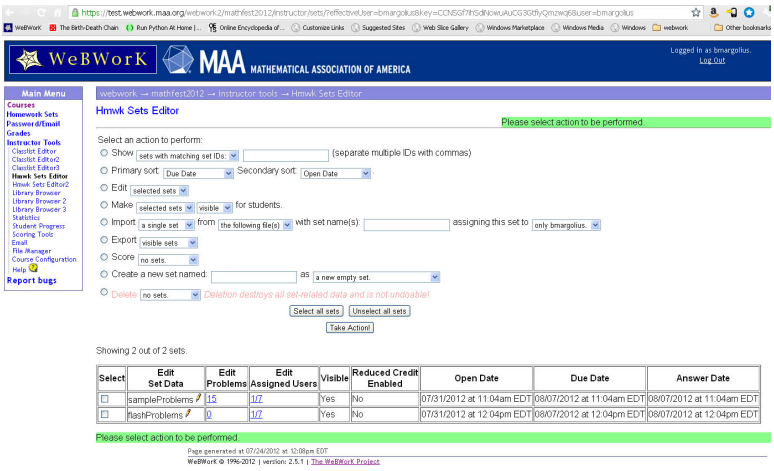


Figure 3: Select "Hmwk Sets Editor".

3. Create a new Homework set.



4. Enter a name for your set in the space beside "Create a new set named:". I called my set "Margolius". Ordinarily you would name your set something like "Homework set 1" or "SeriesConvergence". Once you have chosen and entered a name, press the "Take Action" button.

5. Click the "0" under "Edit Problems" and next to your set to enter the problem set and edit it.

6. Add 10 blank problems. Press the "Save Changes" button to complete this action.

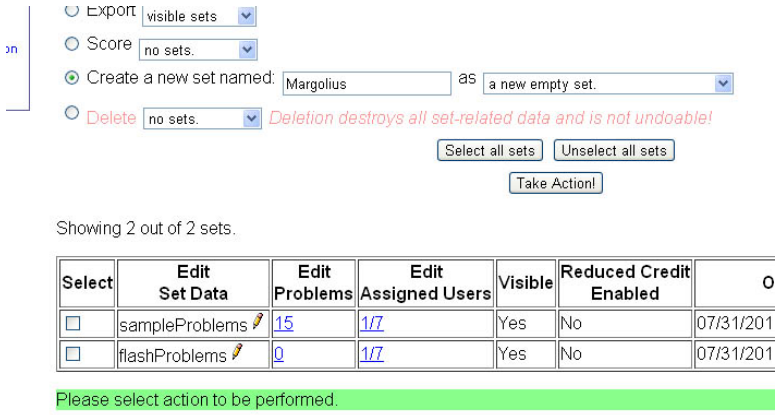
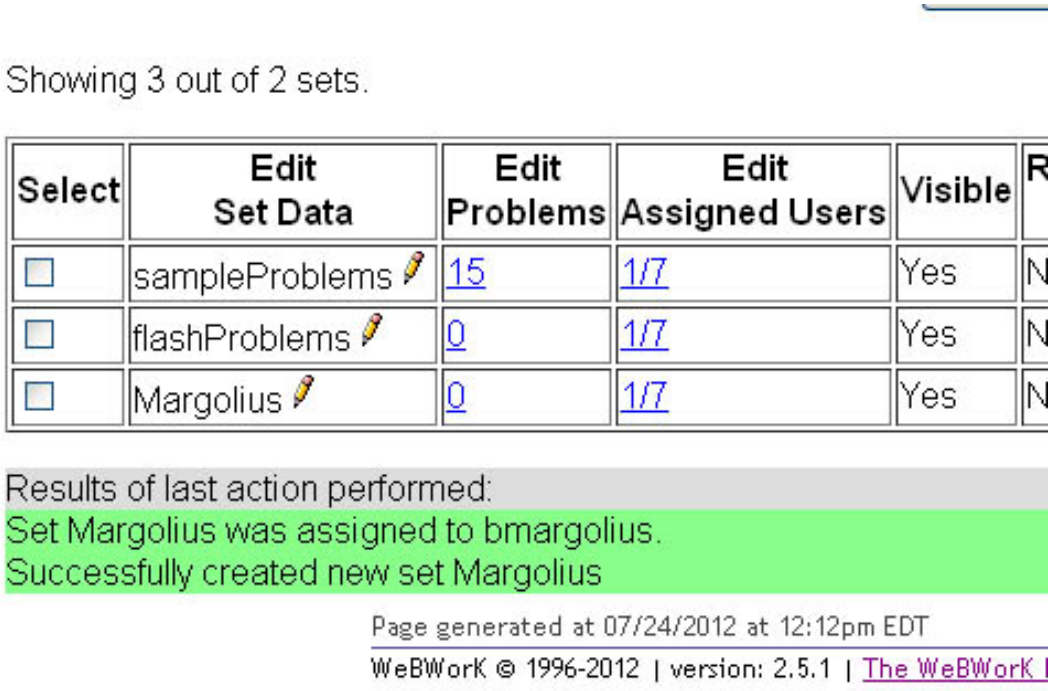


Figure 4: Enter a name for your homework set.



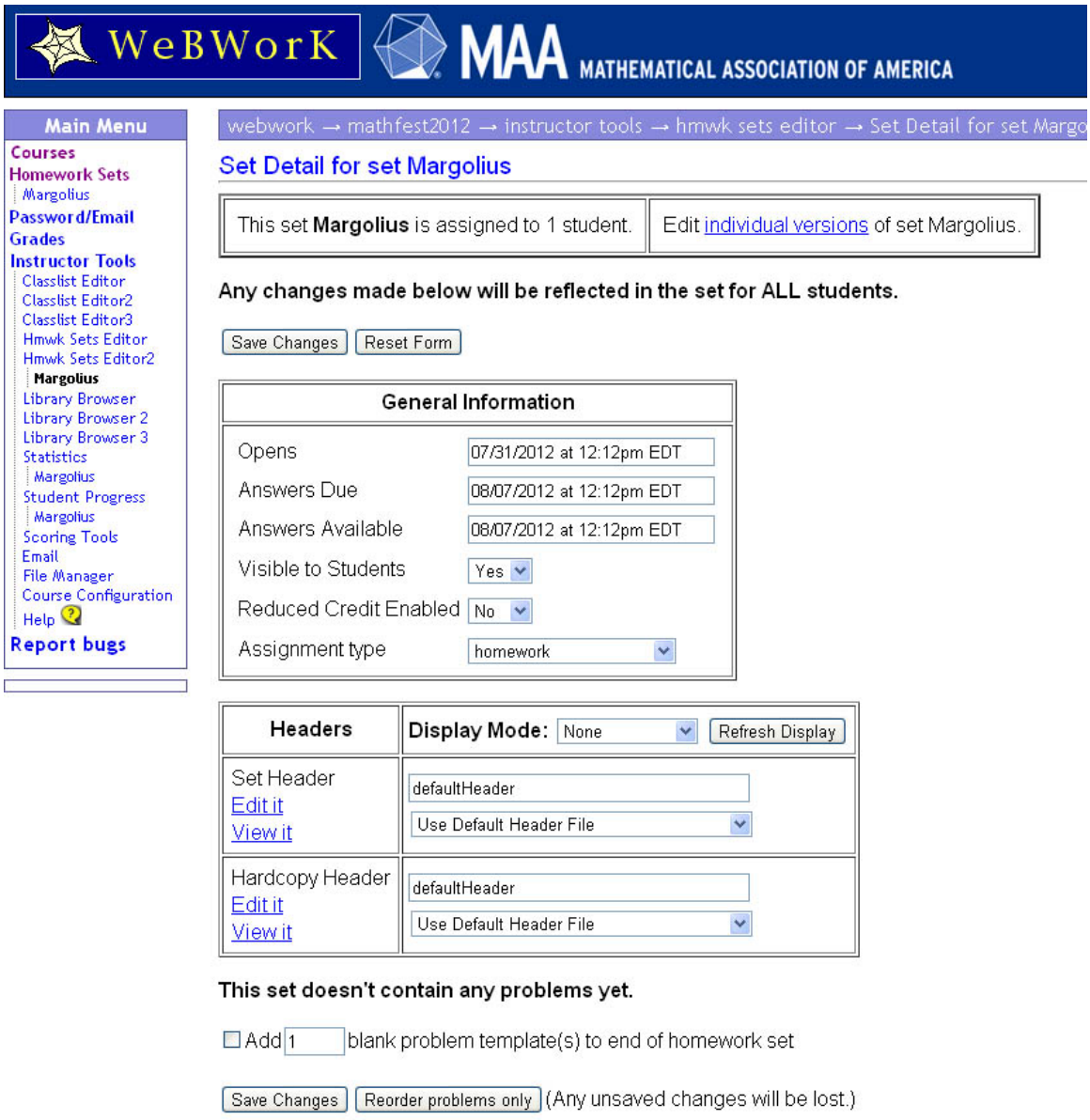
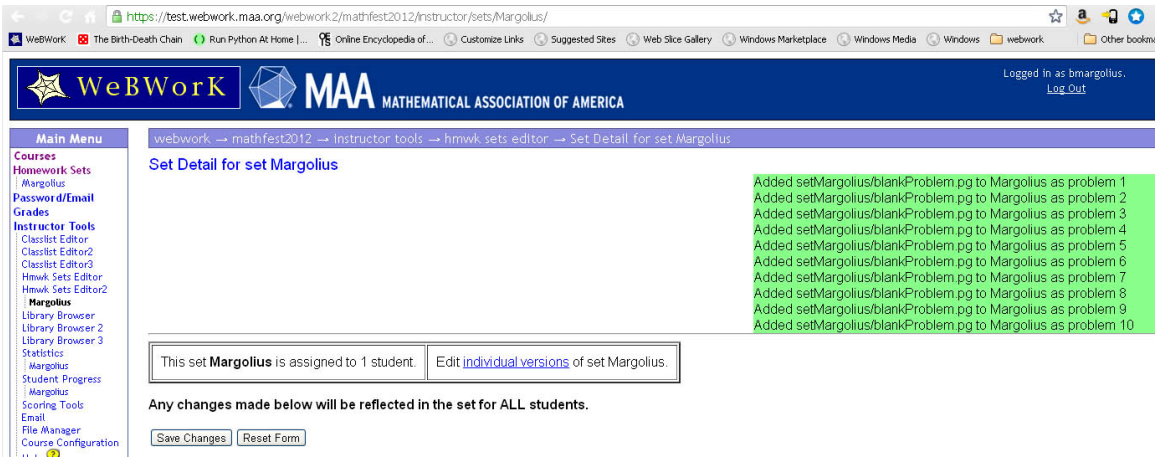


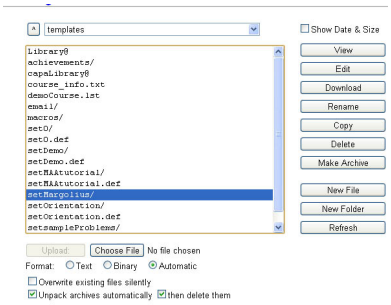
Figure 5: Add 10 blank problems.

7. You have just created your first PG file. A copy of the file "blankProblem.pg" has been placed in a subdirectory called "set(+your homework set name)".

8. Under "Instructor Tools" go to "File Manager".



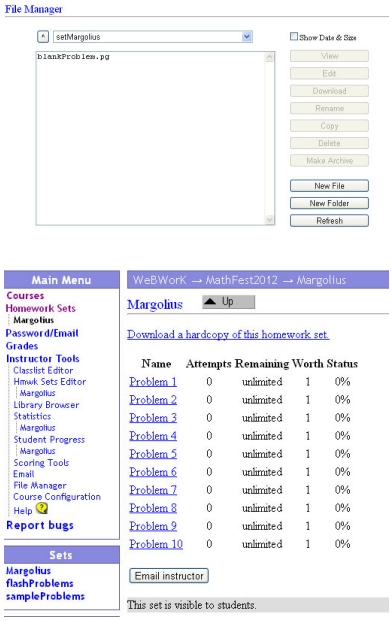
9. Select your folder. In my case, it is "setMargolius". In general you will be looking for a folder called "set" plus the name you chose. I chose "Margolius" so my folder is "setMargolius". Double click on your folder.



10. There is the file I created.

11. To see the actual problem, click on "Homework Sets", then your problem set.

12. Then click one of the problems. Admittedly, this one is a bore. Adding the blank problems though gives us a starting point.



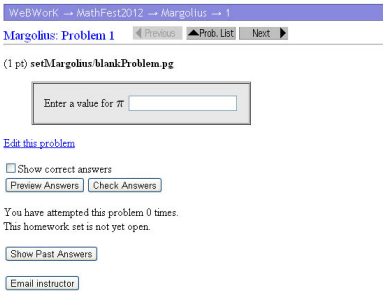
13. We can also add problems in several other ways. We can select problems from the National Problem Library and add them. We can edit the source file line in the homework sets editor to point to another PG file. We can copy problems from the large number of examples on the wiki or we can write our own. Usually if we want to write our own problems, we will want to start with problems from somewhere else.

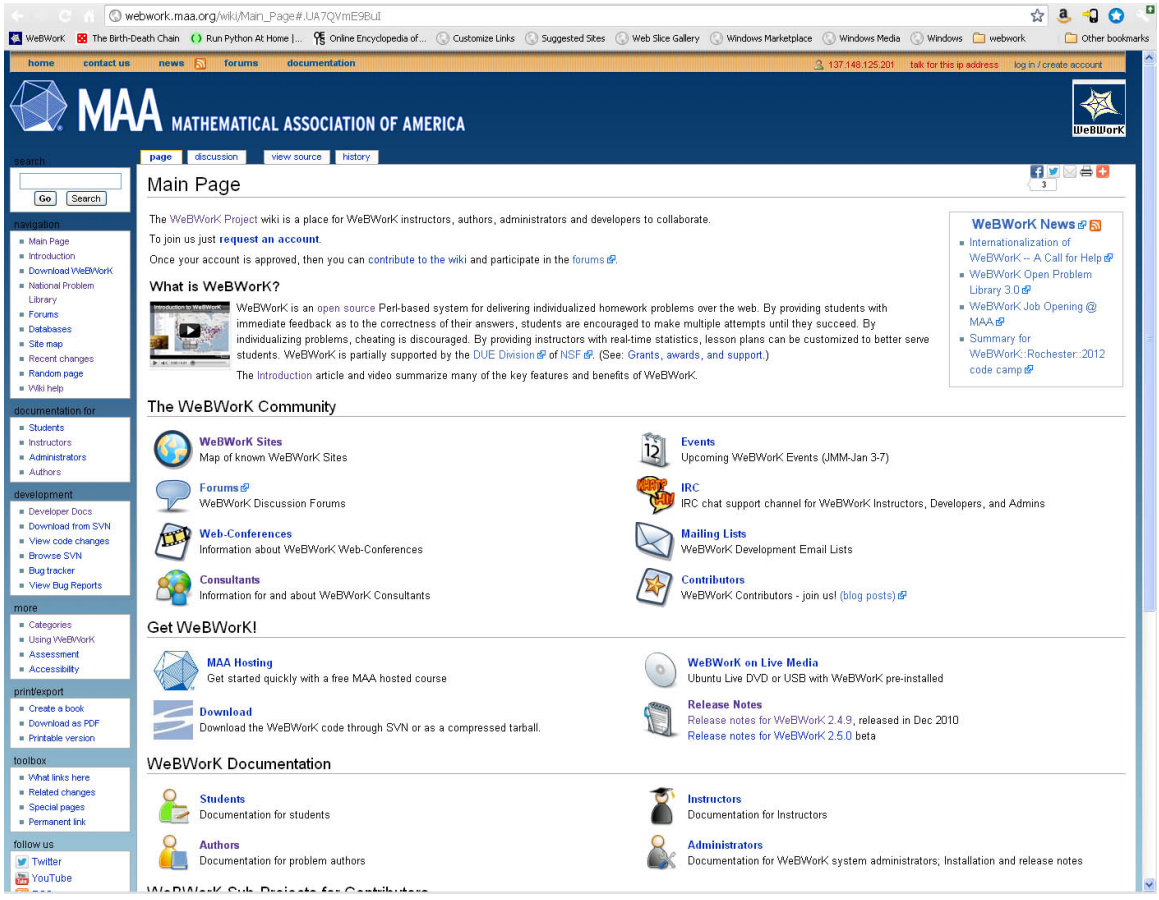
Lesson 2 Finding Problem Examples in the Wiki and Incorporating Them into Problem Sets

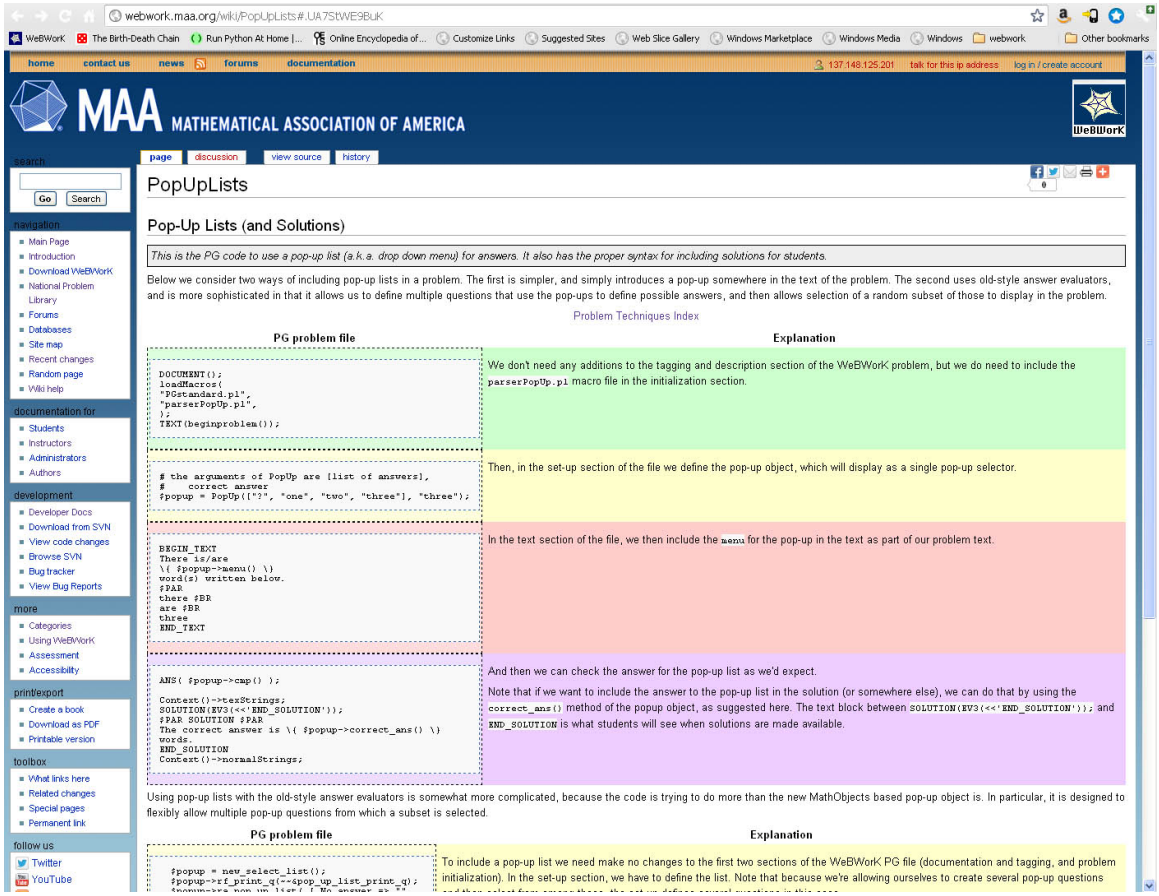
1. Let's create a homework problem with a dropdown list of choices for the correct answer. A good way to do this is to search the WeBWorK MAA wiki for a template. Go to `http://webwork.maa.org/wiki/`.

2. Find the list of problem techniques `http://webwork.maa.org/wiki/Category:Problem_Techniques#`. There are several ways to get there. One way is to select documentation for problem authors, `http://webwork.maa.org/wiki/Category:Authors#` and then click on "Problem Techniques".

3. Now choose "Pop-up lists" and navigate to this page.


WeBWorK → MathFest2012 → Margolius → 1
Margolius: Problem 1
Previous
Prob. List
Next
(1 pt) setMargolius/blankProblem.pg
Enter a value for π
Edit this problem
Show correct answers
Preview Answers
Check Answers
You have attempted this problem 0 times.
This homework set is not yet open.
Show Past Answers
Email instructor


webwork.maa.org/wiki/Main_Page#.UA7QVmE9BuI
home
contact us
news
forums
documentation
MAA
MATHEMATICAL ASSOCIATION OF AMERICA
WeBWorK
page
discussion
view source
history
Main Page
The WeBWorK Project wiki is a place for WeBWorK instructors, authors, administrators and developers to collaborate.
To join us just request an account.
Once your account is approved, then you can contribute to the wiki and participate in the forums.
What is WeBWorK?
WeBWorK is an open source Perl-based system for delivering individualized homework problems over the web. By providing students with immediate feedback as to the correctness of their answers, students are encouraged to make multiple attempts until they succeed. By individualizing problems, cheating is discouraged. By providing instructors with real-time statistics, lesson plans can be customized to better serve students. WeBWorK is partially supported by the DUE Division of NSF. (See: Grants, awards, and support.)
The Introduction article and video summarize many of the key features and benefits of WeBWorK.
WeBWorK News
Internationalization of WeBWorK -- A Call for Help
WeBWorK Open Problem Library 3.0
WeBWorK Job Opening @ MAA
Summary for WeBWorK::Rochester::2012 code camp
The WeBWorK Community
WeBWorK Sites
Map of known WeBWorK Sites
Forums
WeBWorK Discussion Forums
Web-Conferences
Information about WeBWorK Web-Conferences
Consultants
Information for and about WeBWorK Consultants
Events
Upcoming WeBWorK Events (JMM-Jan 3-7)
IRC
IRC chat support channel for WeBWorK Instructors, Developers, and Admins
Mailing Lists
WeBWorK Development Email Lists
Contributors
WeBWorK Contributors - join us! (blog posts)
Get WeBWorK!
MAA Hosting
Get started quickly with a free MAA hosted course
Download
Download the WeBWorK code through SVN or as a compressed tarball.
WeBWorK on Live Media
Ubuntu Live DVD or USB with WeBWorK pre-installed
Release Notes
Release notes for WeBWorK 2.4.9, released in Dec 2010
Release notes for WeBWorK 2.5.0 beta
WeBWorK Documentation
Students
Documentation for students
Authors
Documentation for problem authors
Instructors
Documentation for Instructors
Administrators
Documentation for WeBWorK system administrators; Installation and release notes
search
Go
Search
navigation
Main Page
Introduction
Download WeBWorK
National Problem Library
Forums
Databases
Site map
Recent changes
Random page
Wiki help
documentation for
Students
Instructors
Administrators
Authors
development
Developer Docs
Download from SVN
View code changes
Browse SVN
Bug tracker
View Bug Reports
more
Categories
Using WeBWorK
Assessment
Accessibility
print/export
Create a book
Download as PDF
Printable version
toolbox
What links here
Related changes
Special pages
Permanent link
follow us
Twitter
YouTube



4. In WeBWorK, go to your homework set and open problem 1. Click "Edit1". This will open the blank problem. Notice that the Edit1 link opens the editor in a new window (or tab). This will allow you to navigate between the editor window and the view of the problem as the student will see it. We are going to replace it with the popup problem from the wiki.

5. First let's save it to another file. I have called it "mypopup". I deleted the "local" subdirectory at the front of the file name so it will save "mypopup.pg" in the "setMargolius" directory. I checked the radio buttons to save it and replace the first problem in my set. Now I'll click the "Take Action!" button. So far I have not changed the file itself, only its name.

6. If I go to the homework editor, I can see my "mypopup.pg" file

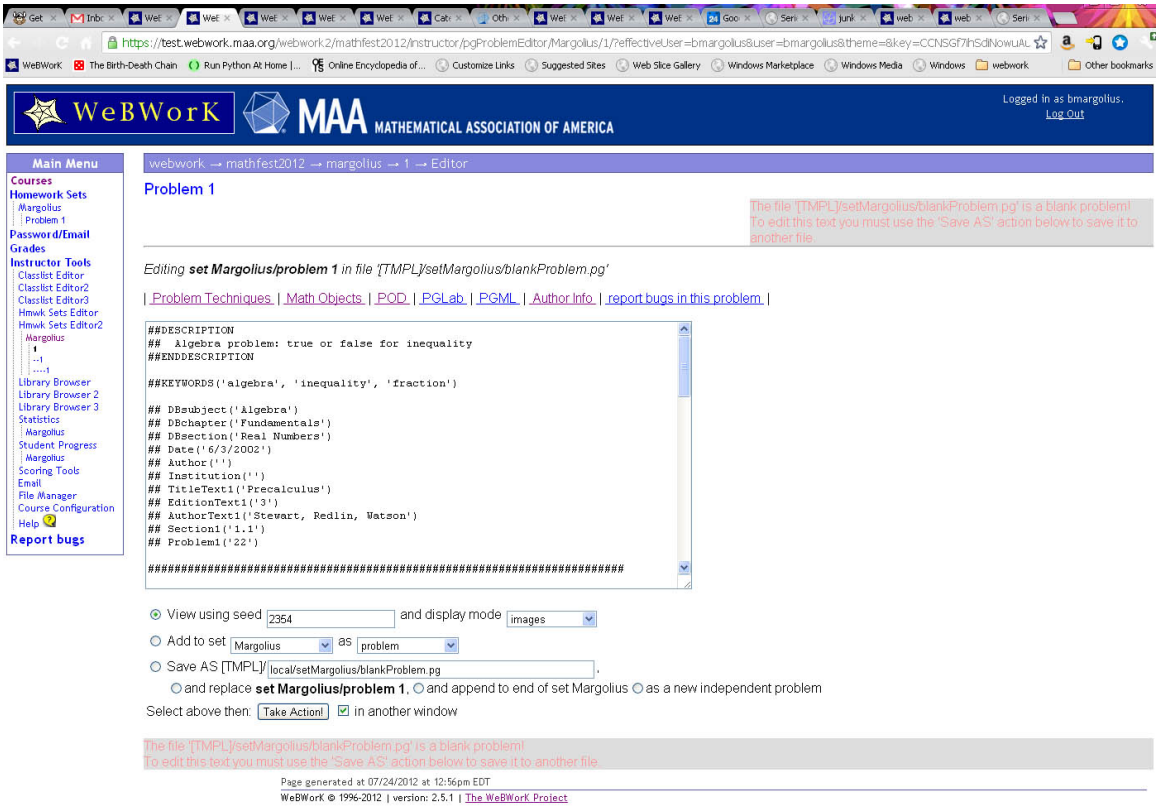
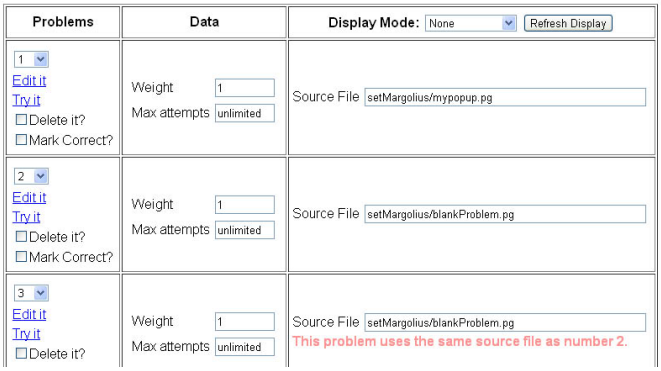


Figure 6: blankProblem PG code.

Figure 7: Rename the blankProblem file.

is the source for problem 1. If I go to the homework set itself,



I'll see it is the same boring problem.

7. Return to editing the problem. Inside the problem editor, select the source code (ctrl or cmd A will do this), then click delete.

8. Copy and paste the code portion of the example from the wiki into your blank problem file.

A complete version of this example is in the "setsampleProblems" directory in the file "popupSimple.pg". It is problem number 2 in the sampleProblems homework set.

A second popup example is given below (this is "mypopup2.pg" in the Margolius problem set).

```
DOCUMENT();
loadMacros(
"PGstandard.pl",
"parserPopUp.pl",
);

TEXT(beginproblem());

# generate a random number between 2 and 6 with a step size of 1
$a = random(2,6,1);
$m = random(2,6,1);


# the arguments of PopUp are [list of answers],
```

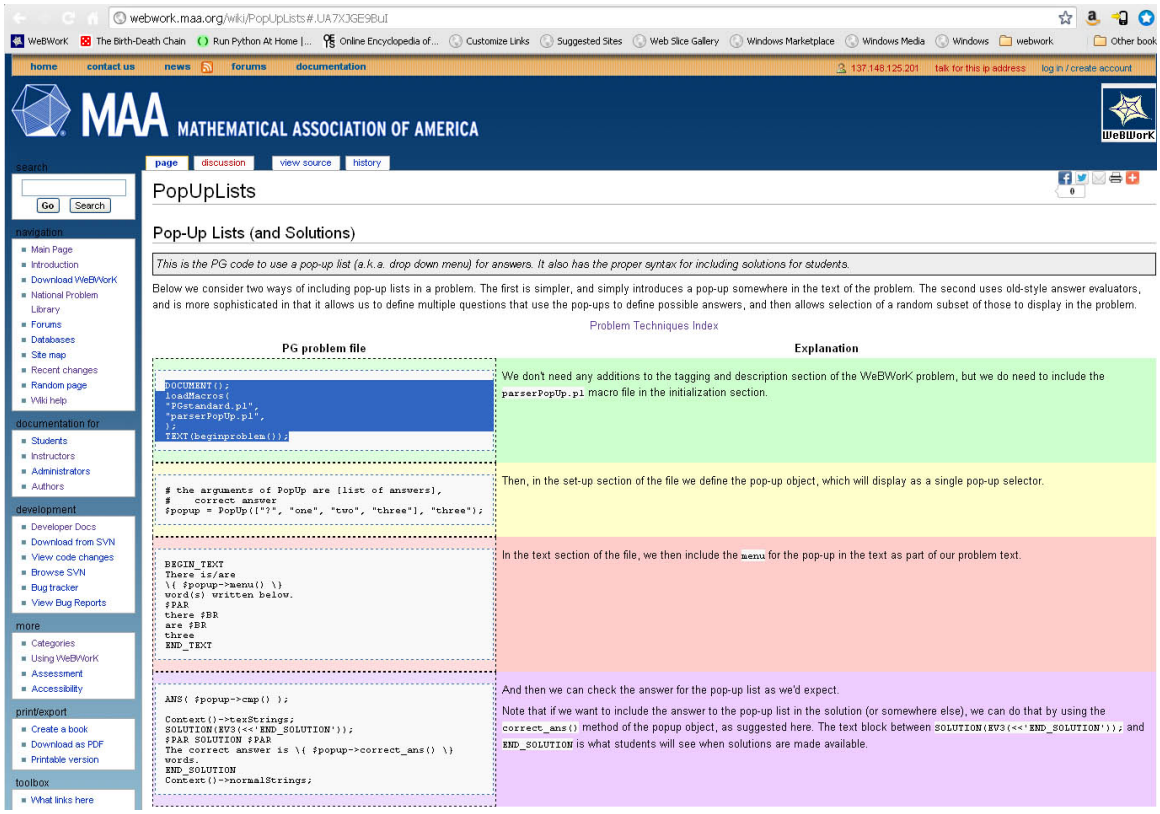


Figure 8: Copy from the wiki.

```
#    correct answer
$popup = PopUp(["?", "converges", "diverges"], "converges");

BEGIN_TEXT
Consider the series
\[\sum_{n=$a}^\infty \frac{(n!)^{$m}}{($m n)!}\]
This series \{ $popup->menu() \}

END_TEXT

ANS( $popup->cmp() );

Context()->texStrings;
BEGIN_SOLUTION
```

```
$PAR SOLUTION $PAR
Using the ratio test, we can show that
\[\lim_{n\to\infty}\left|\frac{a_{n+1}}{a_n}\right|
=\lim_{n\to\infty}\left|\frac{\frac{((n+1)!)^{$m}}
{($m (n+1))!}}{\frac{(n!)^{$m}}{($m n)!}}\right|
=$m^{-$m}<1,\]
therefore the given series  \{ $popup->correct_ans() \}.
END_SOLUTION
Context()->normalStrings;

ENDDOCUMENT();
```

Here is a screenshot of what this problem looks like in WeBWorK. It is problem 2 in the set Margolius.

(1 pt) **setMargolius/mypopup2.pg**

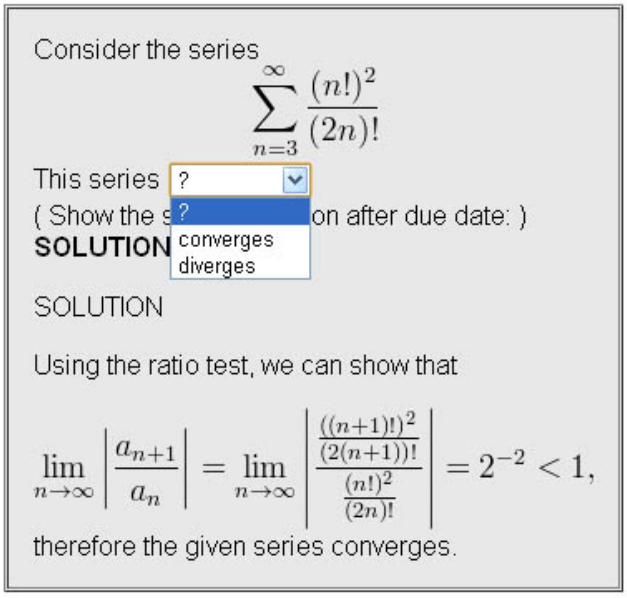


Figure 9: How the student sees the problem.

We will talk more formally about the structure of a WeBWorK problem shortly. Right now, let's look at the parts of this problem. `DOCUMENT();` $\cdots$ `ENDDOCUMENT();` Begins and ends the problem. `DOCUMENT();` and `ENDDOCUMENT();` must appear in every PG problem.

```
loadMacros(
"PGstandard.pl",
"parserPopUp.pl",
);
```

The `loadMacros` function loads a set of helper files called macros. We will need "PGstandard.pl" most of the time. "parserPopUp.pl" is just for problems with popups.

`TEXT(beginproblem())`; Starts the initialization section of the problem. Lines that begin with '#' are comments.

`$a` and `$m` are Perl (and therefore PG) variables. All Perl/PG scalar variables begin with a dollar sign.

```
$a = random(2,6,1);
```

assigns a random integer between 2 and 6 to the parameter `$a`.

```
$popup = PopUp(["?", "converges", "diverges"], "converges");
```

sets up the dropdown list. The possible choices are listed in square brackets. The correct choice ("converges") follows the square brackets but is inside the parens.

`BEGIN_TEXT` $\cdots$ `END_TEXT` Begin and end the text of the problem itself. The text in here is what the student will see.

The text between the delimiters `\[` and `\]` is LaTeX display mode. LaTeX inline mode uses the delimiters `\(` and `\)`. You can learn more about LaTeX here: `http://tobi.oetiker.ch/lshort/lshort.pdf`. So the code

```
\[\sum_{n=$a}^\infty \frac{(n!)^{$m}}{($m n)!}\]
```

generates

$$\sum_{n=2}^{\infty} \frac{(n!)^5}{(5n)!}$$

when $\$a = 2$ and $\$m = 5$.

```
ANS( $popup->cmp() );
```

evaluates the answer and the text between `BEGIN_SOLUTION` and `END_SOLUTION` gives the solution to the problem. The solution is visible to the student after the homework deadline. The solution section can be omitted.

A more detailed solution is given in the "mypopup3.pg" file in the Margolius problem set. The "mypopup3.pg" sample problem illustrates the use of string concatenation and for-loops in generating a more informative solution for the student.

Exercises

1. Create your own drop-down list homework problem in your problem set.
2. Copy another example problem type from the wiki. Get it to function, then modify it. Several sample problem types copied from the wiki are also available in the sampleProblems homework set.
3. Some of the wiki example problems are in convenient files that you can just click on and download. This Riemann sums problem is an example of that: `http://webwork.maa.org/wiki/RiemannSums1#`. Download this problem and place it in your problem set or go to the Subject Area templates and choose another template to download: `http://webwork.maa.org/wiki/SubjectAreaTemplates#`.

3 Using the Library Browser

Lesson 3 Using the Library Browser

You can also find both your own problems and problems from the National Problem Library using the Library Browser.

1. Click the Library Browser link on the left side bar, in the Main Menu. This brings up the browser that lets you look at problems in this course, or in the National Problem Library (NPL).
2. In the Browse row, under the first bold black horizontal line, click the "Local Problems" link to view the problems in this course.
3. Then click the "Select a Problem Collection" selector in the section below the Browse area, and select "setsampleProblems" to see the problems in the sampleProblem problem set. Repeat

these steps and select your problem set instead of "setsampleProblems" to see the problems you have copied or created.

4. Note that you can edit the problems from the Library Browser and you can try them. We won't edit a problem right now.

5. Click on the Library Browser under instructor tools. Note that the National Problem Library button at the top left is depressed.

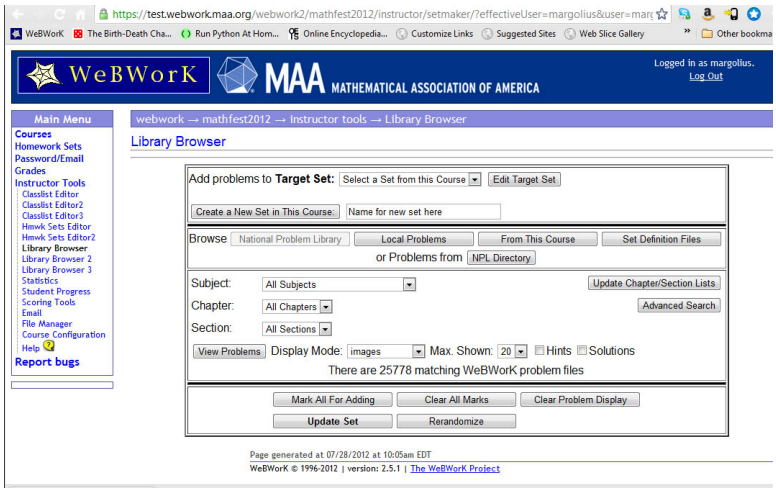


Figure 10: Inside the Library Browser.

6. In basic search, under "Subject" choose "Calculus"; under "Chapter" choose "Infinite sequences and series" and under "Section" choose "The Integral Test and Estimates of Sums". Then click "View Problems".

7. Notice that many of the problems have a turquoise bar across the screen that says "MathObjects version". Select the first MathObjects version probem and any other MathObjects problems that interest you and add them to your problem set. To do this, go to "Add problems to Target Set:" at the top of the screen. Select your set and then scroll through the problems until you see a MathObjects version problem that interests you and check "Add this problem to the target set on the next update". You may select multiple problems if you wish, but pease include the first one shown (Library/ma123DB/set10/s11_3_2.pg).

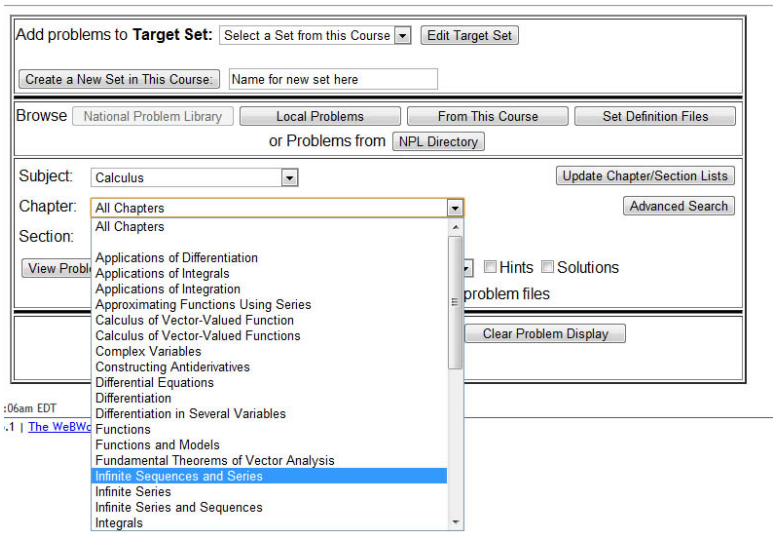


Figure 11: Select "The Integral Test and Estimates of Sums".

We are sticking with the MathObjects version problems because these are coded in the newer style and are the recommended style for problems to code currently and to contribute to the NPL. We want to learn from problems that represent best practices.

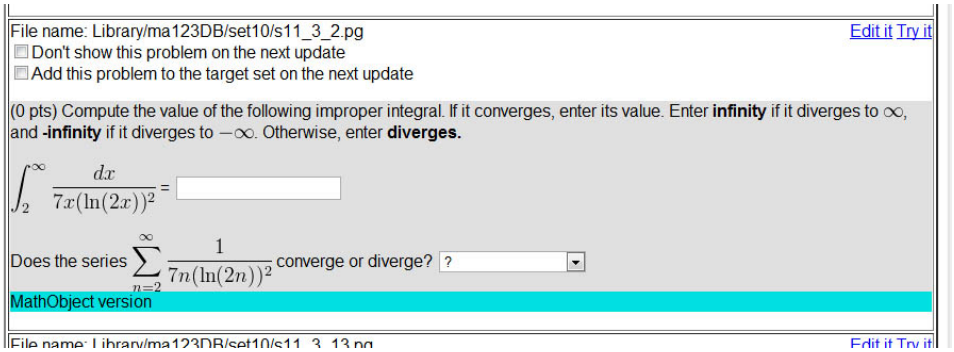
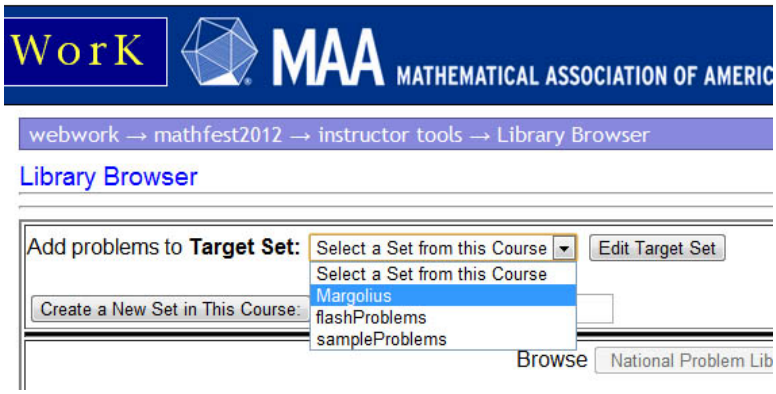


Figure 12: Click the radio button to add this problem to your homework set.

8. Once you have added the problem or problems that interest you, click "Update Set".



Exercises

1. Explore the Library Browser. Look at "Advanced Search" and explore other options in the Library Browser.
2. Choose at least three more MathObjects version problems on any topic you choose and add them to your problem set.

4 Structure of a PG file

Lesson 4 Structure of a PG file

- Tagging info (for the indexing in the National Problem Library)
- Initialization (loading macros, etc.)
- Setup (define parameters, randomization, etc.)
- Main text (the part that gets displayed to students)
- Answer evaluation (checking the submitted answers)
- Solution (optional) and end document (mandatory)

Consider the problem below (given as a screenshot and also with PG code), also provided in your sample problems as problem 5.

```
##KEYWORDS('integral', 'substitution')

## DBsubject('Calculus')
## DBchapter('Integration')
```

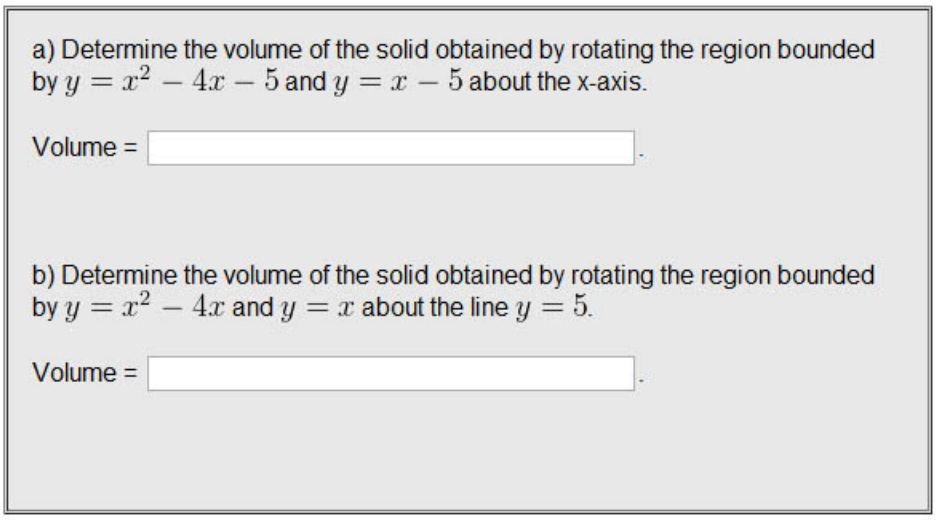


Figure 13: Screenshot of Berney's volume of revolution problem.

```
## DBsection('Integration''Definite Integrals'
   'Area Between Curves''Volumes of Revolution')
## Date('6/18/2010')
## Author('Berney Montavon')
## Institution('Cleveland State University')
## TitleText1('Calculus: Early Transcendentals')
## EditionText1('2nd')
## AuthorText1('Stewart')
## Section1('11')
## Problem1('1')

# This should be the first executable line in the problem.
DOCUMENT();

loadMacros(
     "PGstandard.pl",
     "MathObjects.pl",
     "contextFraction.pl",
);

TEXT(beginproblem());
```

```
$showPartialCorrectAnswers = 1;
Context("Fraction");

$a = random(3,5,1);
$a1=$a+1;
$funca1 = "x^2-$a x -$a1";
$funca2 = "x-$a1";
$funcb1 = "x^2-$a x ";
$funcb2 = "x";
$con =
   Formula("1/30*$a^5+1/3*$a^4+$a^3+4/3*$a^2+5/6*$a+1/5")->reduce;
$volumea = Compute("pi*$con");
$volumeb = $volumea;

BEGIN_TEXT

a) Determine the volume of the solid obtained by rotating
the region bounded by \(y=$funca1\) and \(y=$funca2\)
about the x-axis.
$BR
$BR

Volume = \{ans_rule(45)\}.
$BR
$BR

b) Determine the volume of the solid obtained by rotating
the region bounded by \(y=$funcb1\) and \(y=$funcb2\)
about the line \(y=$a1\).
$BR
$BR

Volume = \{ans_rule(45)\}.
$BR
$BR

END_TEXT
Context()->normalStrings;
```

```
ANS($volumea->cmp);
ANS($volumeb->cmp);

COMMENT('MathObjectversion');
ENDDOCUMENT();
```

4.1 Tagging info

This portion of the problem is the tagging information:

```
##KEYWORDS('integral', 'substitution')

## DBsubject('Calculus')
## DBchapter('Integration')
## DBsection('Integration''Definite Integrals'
  'Area Between Curves''Volumes of Revolution')
## Date('6/18/2010')
## Author('Berney Montavon')
## Institution('Cleveland State University')
## TitleText1('Calculus: Early Transcendentals')
## EditionText1('2nd')
## AuthorText1('Stewart')
## Section1('11')
## Problem1('1')
```

The # at the start of a line marks a comment. Also, if a # is in the middle of a line, the text that follows will be treated as a comment, that is, it will be ignored. The comments in the tagging portion of the problem are required to file a problem in the NPL, specifically `DBsubject`, `DBchapter`, `DBsection` are all required to file a problem in the NPL. Berney wrote the problem in June, 2010. He was a CSU student at the time. The problem is adapted from Stewart's 'Calculus: Early Transcendentals', 2nd edition, section 11 problem 1.

4.2 Initialization

```
# This should be the first executable line in the problem.
DOCUMENT();

loadMacros(
```

```
    "PGstandard.pl",
    "MathObjects.pl",
    "contextFraction.pl",
);

TEXT(beginproblem());
```

PGstandard.pl and MathObjects.pl are Perl macros that should always be loaded. "contextFraction.pl", with Context("Fraction"); will improve the formatting of the answer the student sees. TEXT(beginproblem()); dynamically generates the problem number in the homework set.

4.3 Setup

There are two answer blanks in the problem. Setting $showPartialCorrectAnswers to 1 lets the student know if she got one of the answers correct and which one that was. We set the MathObjects context to numeric and then define some parameters used in the problem. $a is a random integer between 3 and 5. $a1 is one more than $a. We define two functions for the first part of the problem: $x^2-ax-(a+1)$ and $x-(a+1)$. For the second part of the problem, we define another two functions: x^2-ax and x. $volumea is the solution to the first part of the problem. The solution to the second part of the problem is $volumeb which is the same as the solution to the first part, so we have: $volumeb = $volumea;. The single = is defining $volumeb as being equal to $volumea.

The Compute() function determines the kind of MathObject from the Context and from the syntax of its argument. This is usually a string value that is in the form that a student could type. The Compute function also sets the correct answer to be the exact string that it was given, so that if a student were asked to enter a number that matched $a from above and asked to see the correct answer (after the due date), then π will appear as part of the answer, not just a decimal number. This gives you more control over the format of correct answers that are shown to students. For more information see the Introduction to MathObjects in the wiki: http://webwork.maa.org/wiki/Introduction_to_MathObjects#. We are also providing the Introduction to MathObjects pages to you as a handout.

```
$showPartialCorrectAnswers = 1;
Context("Fraction");

$a = random(3,5,1);
$a1=$a+1;
$funca1 = "x^2-$a x -$a1";
$funca2 = "x-$a1";
$funcb1 = "x^2-$a x ";
$funcb2 = "x";
$con =
   Formula("1/30*$a^5+1/3*$a^4+$a^3+4/3*$a^2+5/6*$a+1/5")->reduce;
$volumea = Compute("pi*$con");
$volumeb = $volumea;
```

Note that the range of values for $a is pretty narrow. Don't over randomize when you write your problems. Choose parameter values that you would not mind doing by hand when a student brings a question to you. The code
Formula("1/30 * $a^5 + 1/3 * $a^4 + $a^3 + 4/3 * $a^2 + 5/6 * $a + 1/5")-> reduce;
simplifies the coefficient on π that the student will see as the correct answer after the problem due date. The correct answer will display as a fraction times π, but decimal approximations are still accepted.

4.4 Main text

The main text appears between BEGIN_TEXT $\cdots$ END_TEXT. In this block we leave Perl mode and enter TEXT mode. Within TEXT mode, we can switch to LaTeXmode via \(\) for inline math, or \[\] for displaystylemath. $BR represents a linebreak. \ans_rule(45)\ creates an answer blank of length 45. The Perl variables preceded with a $ sign will be replaced with their values in the display seen by the student.

```
BEGIN_TEXT

a) Determine the volume of the solid obtained by rotating the
region bounded by \(y=$funca1\) and \(y=$funca2\)
about the x-axis.
$BR
$BR

Volume = \{ans_rule(45)\}.
$BR

b) Determine the volume of the solid obtained by rotating the
region bounded by \(y=$funcb1\) and \(y=$funcb2\)
about the line \(y=$a1\).
$BR

Volume = \{ans_rule(45)\}.
$BR

COMMENT('MathObjectversion');
END_TEXT
```

4.5 Answer Evaluation

When using MathObjects, the answer evaluator usually takes the simple form shown here. The evaluators are listed in the order of the answer blanks in the problem (unless we are using a named answer field).

```
Context()->normalStrings;

ANS($volumea->cmp);
ANS($volumeb->cmp);

ENDDOCUMENT();
```

The method $->$`cmp`() is defined for any MathObject. `ANS`(); returns either a 0 or a 1 and takes that result and records it in the database of student scores.

The `COMMENT`(`MathObjectversion`); only shows up for professors in the Library Browser.

Don't forget `ENDDOCUMENT();`. This closes the problem file.

4.6 Solution

The solution section is optional and we haven't included one in this sample problem. You can see examples of the solution section in the problem files mypopup2.pg and mypopup3.pg in the Margolius problem set. Several of the problems in the sampleProblem problem set also have solutions.

Exercises

Create three WeBWorK problem containing the standard sections of a WeBWorK problem: Tagging info, Initialization (loading macros, etc.), Setup (define parameters, randomization, etc.), Main text (the part that gets displayed to students), Answer evaluation (checking the submitted answers), and Solution.

1. Problem 1 in the sample set asks the student to find the derivative of a simple function. Problem 4 asks for the general antiderivative. Choose a simple function and ask the student to find its derivative and general antiderivative. Include at least one randomized parameter.

2. Go to the Function Composition problem technique `http://webwork.maa.org/wiki/ComposingFunctions#`, copy the problem and modify it to ask a similar but different function composition question. Be sure to include the tagging section portion of the problem.

3. Modify the MathObjects library problem you copied: `Library/ma123DB/set10/s11_3_2.pg`. First, give the problem a new name. I have called mine `setMargolius/myIntegralTest.pg`. Select the " Save as" and "Replace set(your set name)/problem (your problem number)" radio buttons and then press the "Take Action!" button. Now you can edit the problem. Change the integral test example. One possibility is to randomize the problem so that it converges in some cases and diverges in others.

5 MathObjects

See MathObjects handout. `http://webwork.maa.org/wiki/Introduction_to_MathObjects#`

Exercises

1. Both `Library/maCalcDB/setSeries8Power/eva8_5a_9.pg` and `Library/Michigan/Chap9Sec5/Q23.pg` are library problems involving the interval of convergence of a power series. The second of these two problems is written in the MathObjects style. Modify `Library/maCalcDB/setSeries8Power/eva8_5a_9.pg` so it is also a MathObjects problem. Begin by saving the problem to a new file.
2. Add a solution to your MathObjects version of `Library/maCalcDB/setSeries8Power/eva8_5a_9.pg`.
3. Write your own power series question using MathObjects.
4. The previous three problems involve the interval context. Earlier in the course, we worked on problems involving the Fraction context. Write a problem that involves one of the other contexts. There are many examples in the NPL that you can use as a starting point.

6 Custom answer checkers

At times, you want WeBWorK to check a student answer with a method different from those available through the default $->$ `cmp`() method of a MathObject. You can write your own answer checking subroutine to override the answer checker routine for the MathObject that is used to check the answer.

Here is a simple example which checks to see if a student's answer is smaller than a given value. The example can be modified to accept answers depending on whatever criteria you wish to apply. The complete code is below. First we have the usual initialization section:

```
DOCUMENT();
```

```
loadMacros(
"PGstandard.pl",
"MathObjects.pl",
);
TEXT(beginproblem());
```

While setting the `Context` to `"Numeric"`, we also set the `tolerance` to zero. We do this because we are basing the correctness of a student's answer on a comparison, and want the full decimal accuracy available in WeBWorK to be used. If we do not do this, the default tolerance in WeBWorK will cause a comparison like $x < 4$ to evaluate to false if $x = 3.999$.

```
$context = Context("Numeric");
$context->flags->set(tolerance=>0);
```

We set an upper bound numerical variable, as a MathObject:

```
$upperBound = Compute('4');
```

We type the problem text and include an answer box for the student to fill:

```
Context()->texStrings;
BEGIN_TEXT
Enter a number which is smaller than \$upperBound: \{ ans_rule(25) \}
END_TEXT
Context()->normalStrings;
```

(We didn't use any TEX above, but you may wish to use it in your own examples.) We end the problem with answer evaluation, with the custom answer checker set as follows:

```
ANS($upperBound->cmp( checker=>sub {
  my ($correct, $student, $ansHash) = @_;
  return $student < $correct;
} ) );

ENDDOCUMENT();
```

Here is what is happening in this code. First, we call `->cmp()` on the MathObject `$upperBound`, while including a custom checker to

override the default behavior of `->cmp()`. The checker routine is a Perl subroutine that takes as its arguments the correct answer, student answer, and answer hash that is being processed in the answer comparison (we are not making use of the answer hash in this example). The return value should be 1 if the student's answer is correct, and 0 otherwise. Because the `->cmp()` method was called on the MathObject `$upperBound`, the value of `$correct` within the subroutine will be set to the value of `$upperBound`. Also, `$student` will be set according to the number input into the answer box by the student. The subroutine returns a value of zero or one depending on the logical comparison `$student < $correct`.

Exercises

1. Create a problem with two answer fields, where the student must input a number smaller than 5 in the first box, and a number greater than 10 in the second box. Check each answer with a separate custom answer checker.

2. Create a problem where a student must input a single value which lies between two given numbers `$lowerBound` and `$upperBound`. [Hint: the answer checker can be called on a numeric MathObject `$ans` with a completely arbitrary value that is not used in the checker subroutine.]

7 Where to Find Things

- `http://webwork.maa.org/wiki/Main_Page`
- Tutorial by Paul Pearson, Fort Lewis College
  `http://webwork.maa.org/w/images/a/ab/WeBWorK_Problem_Authoring_Tutorial.pdf`
  `http://webwork.maa.org/w/images/7/7a/Webwork-PREP-2011-Webconference1-Slic`
  `pdf`
- Problem templates by subject area `http://webwork.maa.org/wiki/SubjectAreaTemplates`
- Index of Problem Techniques
  `http://webwork.maa.org/wiki/IndexOfProblemTechniques`

- WeBWorK documentation http://webwork.maa.org/pod/pg_TRUNK/
- Introduction to LaTeX http://tobi.oetiker.ch/lshort/lshort.pdf
- https://courses.webwork.maa.org/webwork2/gage_course/Knoxsville_MAA_2006_cervone/?login_practice_user=true
- https://courses.webwork.maa.org/webwork2/cervone_course/setAIM-Talk/?login_practice_user=true
- http://webwork.maa.org/wiki/Category:MathObjects