WeBWorK User Guide for PCC Instructors

Alex Jordan

alex.jordan@pcc.edu

September 15, 2015

This document is meant to serve two purposes. For PCC faculty who are already familiar with WeBWorK, this is meant to be a reference. From the table of contents you may hyperlink to any topic. For PCC faculty who are new to WeBWorK, the document can be used linearly as you set up your course.

Contents

Overview

WeBWorK is accessed through a web browser. Its primary purpose is for your students to submit mathematical answers to homework problems. It has several other uses including timed exams, placement testing, data-gathering, and general collection of (not necessarily mathematical) student responses.

You will have access to several libraries of problems, each problem of which is coded in the form of a plain text file. A *homework set* is a list of problems from these libraries together with a due date and certain other options. When a homework set has been created, you assign it to your students, or subsets of your students. WeBWorK keeps track of all student input and their scores. Note that a *homework set* does not need to literally be a homework set. It could be a review set, an exam, a survey, reading material, etc.

Your role[1] is to:

1. Add your students to the course and choose some initial configuration settings.

2. Create the problem sets. I recommend getting this all laid out before the term starts. It is a one-day investment that will reduce your workload during the term. You will have complete flexibility to change anything later. For future versions of the same course, this process is very fast; you can copy an old setup and just change dates.

3. Respond to student emails for help. WeBWorK will allow students to email you rather effortlessly for help. Since the volume of email that this creates can get large, I recommend setting up an email filter to separate any incoming message with [WWfeedback] in the subject header. When you have time and are in the right mindset to respond to student emails, you can go through this pile more efficiently.

4. Put out occasional fires. Maybe a student forgets their password. Maybe you want to grant an extension to an assignment. Maybe there is a code issue with a problem that you have assigned (in which case you may need to call for help.)

5. Gather the grades from WeBWorK and import them into your grade book.

If you are brand new to WeBWorK then I suggest progressing in a straight line through this guide. You will know when it is OK to skip something. If you have more than one section of a course, then focus on one section first. You will be able to quickly copy your setup once the first section is in good shape. If you are not new to WeBWorK then consider this document to be a reference guide.

Logging On and Passwords

Unless you have a special case,[2] your course is located at (using all lowercase letters):

---

[1]More complicated roles are possible.
[2]Some special courses may have a different structure to their address.

<div align="center">

`http://webwork.pcc.edu/webwork2/aaabbb-ccccc-lastname`

or at

`http://webwork.pcc.edu/webwork2/aaabbb-astname-term`

</div>

where aaa is your subject code (mth, cs, etc.), bbb is your course number (060, 111, etc.), xxxxx is your CRN, `lastname` is your last name, and `term` is something like `f15`. You can ask to remove your last name from the CRN-specific address, but having it there can make it easy to find your own courses from among the list at `webwork.pcc.edu`. Make a bookmark with this link at your work station—you will need to come here often. For your students, you should make a link in MyPCC under the **Links** section. (Don't make them have to directly type the awkward web address.) It's also possible that your course is listed at http://webwork.pcc.edu/webwork2/, but it may be hidden.

In most cases, everyone's login should be their complete PCC email prefix, using *all lowercase* letters. If your email is `jane.doe15@pcc.edu`, your login should be `jane.doe15`. The default password should be your G-number, using *a capital G*. It might be your old password if you have used WeBWorK before. If you cannot log on, contact Alex Jordan (SY) or Carl Yao (SE), each of whom has administrative access to your account. Everyone (especially instructors and TAs) should change their password the first time that they log in. This is done using the **Password/Email** item under WeBWorK's main menu.

Initial Setup

When you first receive your WeBWorK course, there are a few items to initialize. On the home screen when you log in, you will see **Course Info** off to the right. You should **[edit]** this, providing all students with some basic course information. You may use html tags to improve the look of this section or link to other web resources, but that's not necessary. (This is all stored in your course's **File Manager** as `course_info.txt` if you would like to access that file directly.)

Next, there are some options you might want to tend to in the **Course Configuration** menu. Browse through all five tabs in the **Course Configuration** menu. All options should be set to something reasonable by default. If you understand what an option does, feel free to change it. Of course if you do not understand an option, leave it alone or feel free to ask about it. The most common items you may want to change settings for are:

Enable Conditional Release

If you would like to make it possible to conditionally release homework assignments, you can enable this option. For example, this can be used to make it so that Assignment #2 is not available until a student has say a 90% on Assignment #1, etc.

Course Achievements

WeBWorK has a feature called Course Achievements. This is a way to award achievement points (separate from regular homework points), badges, levels, and special items to students to make the whole experience more like a game. For many students, this 'gamification' of homework is motivational. If you like, enable Course Achievements in the **Course Configuration** menu. Once you do this, your navigation panel will have an **Achievements Editor** in the instructor tools. Also, all users will have an **Achievements** menu to monitor their achievements. To make use of this feature, you need to import achievements and assign them to your students. This is discussed in more detail in the Gamification section.

Enabling MathView

This is a palette tools that may make it easier for students to enter complicated expressions. If it is enabled, there will be an icon next to each answer blank. Clicking this icon will open an applet window that allows students to construct an expression with radicals, fractions, and more using a palette similar to MathType (but with few complications). MathView is enabled by default. If a student has a MathView window open, then they can see the "pretty-print" version of their answer as they type it. For example, as they type (x+1)/x, they will see this sequence appear character by character:

$$(\qquad (x \qquad (x+ \qquad (x+1 \qquad (x+1) \qquad \dfrac{x+1}{} \qquad \dfrac{x+1}{x}$$

Reduced Scoring for Late Work

Your assignments will each have a due date/time. If you like, you may allow students to work beyond the due date/time for reduced credit. If you plan to use this for any homework sets

- Enable Reduced Scoring

- Set the Length of Reduced Scoring Period in minutes

- Set the Value of work done in Reduced Scoring Period

Later when you create an assignment, you will *still* need to enable reduced scoring for that particular assignment within the **Hmwk Sets Editor**, since it is possible to have some assignments use reduced scoring and other assignments not use it. You can read about this in the section on creating homework sets.

Warning: If you use this feature, there is currently an unfortunate arrangement. The displayed due date will be the date and time beyond which students cannot submit *any* more answers. The reduced credit period begins *prior* to this. WeBWorK *does* explain this to students alongside the due date, but this arrangement is potentially confusing.

Show Me Another Button

If you enable this feature, then you will have the ability to make it so that a student can see a re-randomized version of a problem. In the new version, you can make it so the student can see the answer, the walk-through solution (if there is one), the hints (if there are any), and have access to the **Check Answers** button. You will have the ability to control which problems this applies to. For more information, see the section on Show Me Another. This feature is supposed to serve as an analog to a solutions manual or textbook examples.

If you think you will use this feature, you should go to the section on Show Me Another now and read about it. There may be some configuration settings that you would want to set *before* you import or build any homework sets.

Adding/Dropping Students

In the future, enrollment of students may be automated through Banner. For now, it falls upon you to enroll students and enter their information.

Adding Many Students at Once

You may add a full roster of students to your WeBWorK course all at once with minimal hassle if you do the following little dance with MyPCC and Excel. I recommend waiting until the weekend before classes start to follow the procedure below to minimize the hassle that comes with all the flux of enrollments right before a term begins. The goal is to create a plaintext `.csv` (comma separated value) file that has all the pertinent information in it. Each line of the file will have a student's information with no spaces, organized as:

```
G-num,L-name,F-name,status,comment,section,recitation,email,login,password,permission
```

If you understand all this so far then you may not need any further instruction. But more detailed step-by-step instructions follow. If you have trouble, send Alex Jordan your full class roster in the form of the `.xls` file that you can download from MyPCC.

1. In MyPCC, go to your **Summary Class List**. At the bottom, there is a link to **Download Roster**. Click on this and open the file with Excel.

2. Delete the first four columns so that G-numbers slide over to column A. Also delete the top two rows so that the remaining top row is actual student data. Note that you may have wait-listed students at the bottom. It's your choice to enroll them in WeBWorK or not. You can always delete them (and any no-show students) later.

3. Highlight column C, and **Insert** columns until the student email addresses are in column H. This should leave column A with G-numbers, B with names, several blank columns, H with

email addresses, and more beyond that. Clear the contents of all columns beyond the email addresses.

4. Highlight the column of names. In the **Data** menu, select **Text to Columns**. Choose the **Delimited** option. Add **Comma** to the list of delimiters, and click **Finish**. Column B should now have your students' *last* names, and column C should have their *first* names and middle initials.

5. Fill column D with the letter C. (You can enter a single C at the top of the column and drag it down the column's length.) The C stands for Currently enrolled. If a student is auditing, an A belongs here instead.

6. Columns E is for a comment about the student. I usually leave this blank, but make a note if a student is only on a wait list at that point. Column F is for noting the section. If there is only one CRN, I leave this blank. If there are multiple CRNs, I typically enter something like "42483 TuTh morning", and this will allow you to sort students by section and assign due dates differently to different sections. Column G is for recitation. At PCC, I don't see a need to enter anything here. But if you did, it would serve the same purpose as the section column, and just allow for multiple ways to sort.

7. Copy column H (which should still have email addresses) to column I. Highlight column I and add the @ symbol to the list of delimiters by once again using **Data→Text to Columns**. Clear all the `pcc.edu`s from column J. Now column H should *still* have email addresses, and column I should have their user logins.

8. Leave column J blank. (This is where you could use a password hash from some earlier course. Leaving it blanks makes it so their student ID (G number) will be their initial password. Fill column K with the number 0. (This sets their permission level to `student` as opposed to say `professor`.)

9. Save the Excel document as a `.csv` file. Excel may give you warnings as you do this about losing data, but don't worry.

If you are putting multiple CRNs into one WeBWorK course, then you can either repeat this process for each CRN, or you can make one big CSV file with all the courses. Just remember to distinguish the students by section using the section column.

Your `.csv` file needs to be renamed to have a `.lst` (that's ℓst, not 1st) extension. If you have trouble doing that in your operating system, you can also rename the file within WeBWorK. Use WeBWorK's **File Manager** to upload your `.lst` file straight to the `templates` directory. Check that it really does have the `.lst` extension (and if not, use the **Rename** tool). Then in the **Classlist Editor**, use **Import users from file** to add all of these students to your WeBWorK course. If any problem sets have already been created or imported, then you should now assign them to all of these new students by using the **Instructor Tools**.

Adding Students One (or a Small Number) at a Time

In the **Classlist Editor** there is a tab for adding students by hand. Fill in as much information as you can, but the minimum that is needed is a **Login Name**, a **Student ID**, and an **Email Address**.

Initial passwords will be the **Student ID**s. As you individually add students this way, you may choose existing assignments to assign to these new students, or you can do this later using **Instructor Tools**.

If you are using the Achievements in your class, be sure to assign achievements to these newly-added users—see the section on assigning achievements to lately-added students.

The Orientation Assignment

There is an Orientation assignment for your students to take.[3] This orientation tries to address common issues that students have when using WeBWorK. Once you have your students enrolled, import the Orientation assignment by following the instructions in the Orientation section. Or if you are progressing through this document linearly, you can wait until you get there. *Be sure to actually assign the Orientation to all students.* Simply importing the Orientation is not enough.

Removing Students

Before the course begins, you may remove students by deleting them in the **Classlist Editor**. After a course has begun, it is strongly recommended that you *drop* them, instead of *deleting* them. To *drop* a student, select their checkbox from the **Classlist Editor** and **Edit selected users**. Change their status. By dropping rather than deleting, the student's data is preserved, just in case it is needed.

Guest Accounts

You can set up your course so that guests may log in. Guests may view problems from homework sets that have been assigned to the guest IDs. Their data is not stored. If the due date has not yet passed, guests may *not* check their answers for correctness. If you would like to have guest accounts enabled, go to the **Classlist Editor** and import users from the file `demoCourse.lst`. This creates "students" in your course list named practice1, practice2,. . .,practice9. If such a "student" is part of your roster, then the login screen will have a **Guest** login button. If you later wish to disable guest accounts, simply delete these "students" from the roster.

Making a New Problem Set

Problem sets can either be imported from an earlier course or made from scratch. If your course came with problem sets that you would like to import[4], you may wish to skip ahead to the section on importing assignments. Brand new users teaching MTH 8/20/60/61/62/63/65/70/95/111/243 should skip ahead and import the default assignments that are available (and then modify them if

---

[3]If you are not using WeBWorK as a homework platform, there is probably no need for this.

[4]Of course, these may have an ordering that is not in line with your syllabus. Still, they are sets of vetted problems. So it might be worth importing them and learning how to modify these sets to meet your own needs.

that is desired). If you have run this course before and wish to copy your old problem sets, you should skip ahead to the section on exporting assignments.

To create a new problem set from scratch, in the **Hmwk Sets Editor** or the **Library Browser**, you may **Create** a new set and name it whatever you would like, just no periods or weird characters. You *can* use spaces though For example, `Assignment 1` or `Chapter 3 Review` are good names.

Add Problems

To add problems to an assignment, go into the **Library Browser**. Select the **Target Set** from the dropdown menu. There are several sources of problems for you to browse.

For MTH 60–70 select the **Basic Algebra** library. From the **Select a Problem Collection** dropdown menu, choose the relevant topic. These problems were written by Chris Hughes, Alex Jordan, and Carl Yao during summer 2013. Each problem has a walk-through solution, uses high-quality WeBWorK coding techniques, and was written with accessibility standards in mind.

For MTH 20 select the **Basic Math** library. These problems were coded by Carl Yao mostly over Fall 2013. Each problem has a walk-through solution, uses high-quality WeBWorK coding techniques, and was written with accessibility standards in mind.

For MTH 111 select the **College Algebra** library. These problems were coded by Alex Jordan mostly over Summer 2014. Each problem has a walk-through solution, uses high-quality WeBWorK coding techniques, and was written with accessibility standards in mind.

For MTH 95 select the **Math 95** library. These problems are generally not from a textbook and use randomization. These problems are currently being improved upon to reach the same standards as the 60/65 problems. MTH 95 overlaps some with MTH 60/65, and you will find usable problems in the Precollege Algebra library too. During Fall 2015, Carl Yao and Heiko Spoddeck will work to improve the quality of the MTH 95 problems.

For MTH 243 select the **Statistics** library. These problems are generally not from a textbook and use randomization. These problems are currently being improved upon to reach the same standards as the 60/65 problems.

For physics problems take a look in the **CAPA** library. CAPA is an older online homework platform for physics problems, and some time ago an automated process converted CAPA problems to a state where WeBWorK can use them. These problems do not necessarily use the best coding techniques, and they often come with pictures that currently do not have good alt tags for accessibility.

For *any* course, you may look for problems in the **Open Problem Library**, formerly known as the **National Problem Library**. These are sortable with a simple search or the **Advanced Search**. Lastly try looking at the **OPL Directory**, which organizes the OPL library according to who contributed the problems and what their indexing system was. The problems from Union College and the College of Idaho are notably well-written, but there are many institutions that have written good problems.

Click **View Problems** to see the problems. The default is to see 20 at a time, but you can change this. You can click the re-randomize icon for each problem to see how its randomization behaves. Use the **Add** button to add problems to the problem set you have targeted.

Set Description

In the **Hmwk Sets Editor**, after clicking in the numerical link for your set under **Edit Problems**, there is a field to give a brief description of the set. When users hover their mouse over the name of the problem set, this description appears as a tooltip. For example, you could *name* a homework set as *Chapter20,* and give it a *description* that is a little more detailed, like *Hypothesis Testing for Population Proportions*.

Assign the Set to Students

Use the **Instructor Tools** to assign the set to students. At first, the problem set will have an opening date in the future, so there is no concern that students will log on and begin working before you have completed the setup. Of course, if you have not yet enrolled your students, you will need to come back to do this later.

Do the Assignment Yourself

This isn't really necessary. However, it is a good idea. This way, you will encounter issues before your students do. (If you find an issue with a problem, you can edit it, replace it with one that works, or ask someone who knows the WeBWorK programming language to look into it.)

Take notes concerning any input issues the problems have. Will students need to be reminded how to enter π or ∞? Will they need to know that they should answer in function notation, as with $P(t) = 1.1^t$ rather than just 1.1^t? Well-written problems will make these issues clear. Take notes on issues like this that you come across and you can put them into the assignment header later.

Set Attempts

By default, WeBWorK will allow students unlimited attempts for each problem (while keeping track of how many tries a student uses and recording their incorrect answers). In the **Hmwk Sets Editor**, click the numerical link under the column **Edit Problems** corresponding to the set in question. You probably want to change the **Display Mode** for the **Problems** to **MathJax** or **Images**, and then click **Refresh Display**. Problem by problem, you may change the number of allowed attempts.

For multiple choice questions, you may only want to allow one or two attempts. For matching questions, maybe more. I recommend allowing unlimited attempts for all other problems. Reasons for this are discussed below in the Good Ideas section.

Set Show Me Another Threshold

If you have enabled the Show Me Another feature, then for each problem you can indicate how many attempts a student must use before they gain access to the **Show Me Another** button. Setting this to 0 will give students immediate access to the **SMA** button for that problem. Setting this to −1 is the default, and this means that the student will never have access to the button.

Assignment Header

When a student opens an assignment, they see some text off to the right. The intent is to give information to the student that are specific to that assignment. You can tell students which chapters from the book are relevant, give links to places on the Internet, give tips on entering answers, etc. If you do nothing, then some default text will appear here. Best practice is to replace this text with your own instructions and warnings.

Unfortunately, the source file that is used to print these messages is written in the language that WeBWorK uses for generating problems, which is a combination of Perl, LaTeX, and WeBWorK-specific elements. For the uninitiated, writing this section from scratch would be difficult.

Your **File Manager** has a file called `ASimpleScreenHeaderFile.pg`. In the **Hmwk Sets Editor**, click the numerical link under the column **Edit Problems** corresponding to the set in question. In the big **Headers** box, use the drop-down menu to select `ASimpleScreenHeaderFile.pg`. You should then **Edit it** to make it relevant to the assignment you just created. After you have made your alterations, use the **New Version** tab to save the file with a new name, and be sure to check the radio button that replaces the set's header file. Now students will see this nice header when they open this assignment.

One advantage of exporting and importing old assignments is that you don't generally repeat the process of setting up header files—you just reuse your old header files.

Hardcopy Header

A large number of students will put their assignments into `.pdf` files, to work with offline, and then return to enter answers later. A *different* header file called the *hardcopy header* is used for printed versions. The lazy thing to do is to use the same screen set header that you just made. For technical reasons, this is not ideal. Formatting will be off—a minor concern. Somewhat more importantly, the characters <, >, {, and } will not show up correctly. Even worse things may happen if you were using characters like $, ^, _, %, etc.

If you want to do things right, set the hardcopy header to `ASimpleHardCopyHeaderFile.pg` and then edit it in a similar way to how you created the screen header file. (Or you can just use your header as a hardcopy header and *probably* nothing serious will go wrong.)

Set the Dates

In the **Hmwk Sets Editor**, **Edit** the set that you just put together. You may set the date that the problem set is available to students, the date that it is due, and the date that answers are made available to students.

Be careful with setting the due time. It is tempting to use midnight, but don't use 12:00AM, since many students will confuse this with noon. Also, if you set a due time to 11:59PM Tuesday in the fall, then the daylight savings changeover will make the due date become 12:59AM Wednesday, and some students will erroneously think that the assignment is due Wednesday at midnight. One last consideration—as a due date/time approaches, many students will log on. If this simultaneously happens across many courses, it has the potential to overload the server. For that reason, consider setting a due date/time that is in the middle of a weekday, when tech support is still available to get the server back up.

I recommend that you set the **answer date** (when answers and solutions become visible to students) to one or two days after the **due date**. This is because you may want to grant an extension the day after an assignment was originally due, and you would not want the answers to have been made available in the meantime.

Default due times and time lags between the three critical times for an assignment may be set in the **Course Configuration**.

Reduced Credit Period

If you like, you may set a time period at the end of an assignment's window in which students only receive partial credit. The idea is that you really have an earlier due date in mind, but you are granting a penalized grace period for late submissions. If you plan to use this, go into the **Course Configuration** and set both the duration of the reduced credit period, and the fraction at which you will offer reduced credit. You will then need to enable the reduced credit period for that particular assignment within the **Hmwk Sets Editor**. *Warning:* If say the due date is set to Friday and you set a 24 hour reduced credit period, then students will see a due date of Friday but only have until *Thursday* to get full credit. This is explained to students alongside their due dates, but it is nevertheless potentially confusing.

Quizzes/Exams

With WeBWorK you can create quizzes that students can take online, proctored or not. You can also use WeBWorK to generate quizzes *that are possibly different for each student* and print them off or have them taken at a computer station. Here is an outline of how to create a quiz/exam.

1. Create a 'homework set' that will become the quiz; give it an appropriate name like Quiz1. Keep in mind that the set may draw randomly from a "problem group", discussed in Problem

[Groups](#). (For example, problem number 5 on the exam could be a randomly chosen problem from a list of problems all having to do with solving a linear equation.)

2. In the set details for that set (in the **Hmwk Sets Editor** click on the number for that set under the column **Edit Problems**), use a drop down menu to turn the HW assignment into a quiz (either proctored or not) and save changes.

3. Set a bunch of parameters for the quiz that appear, discussed below.

4. If the quiz is proctored, arrange for a computer lab and proctor. Proctors either enter their own login information into each student's screen one at a time, or they can verbally give a password to all of the students in the room that will let them start.

To help you understand the multitude of options for a quiz, consider this scenario that an instructor may have put in place. Some of the chosen options may seem silly, but this example is just for demonstration.

> The quiz becomes available on Monday morning at 12:00AM and the due date is Friday at 11:59PM. Each student is allowed to take up to 3 versions of the quiz. Each new version will use new random seeds, so the questions will be different. Once a student starts a version, they have one hour to complete it. After the due date, an exam may not be started, but had a student started just before the due time, they do have a full hour. Each time a student is working on a version of the quiz, they may submit it twice for a grade (allowing them to make corrections after the first submission). In any 24 hour time interval, the student may start at most 2 versions. Problems are ordered randomly from student to student. Only three problems at a time are shown on each screen. Once a student has no remaining attempts or time for a version, they are granted access to their score for that version of the entire quiz. However, they do not get to see their scores or answers for the individual questions until after the due date.

If you are new to WeBWorK Gateway quizzes, I recommend that you try taking your own quiz and experimenting to check that you have all of the options the way you wish them to be.

Problem Groups

It is possible to define a collection of related problems, say all having to do with "section 2.4", and then have your quiz pull problems randomly from that set. (Unfortunately this feature is currently only available for *quizzes*, not regular homework assignments.) This is especially useful if you want each student's quiz to be different from their neighbor's.

This walkthrough explains how to do this for such a problem group. First, create an 'assignment' consisting of all of the problems that you would like to group together. There is no need to assign this set to anyone. For an example, suppose we name this set `section2_4problems`.

Now in the assignment that will be the assigned quiz, in the **Hmwk Sets Editor**, click on the number for this assignment in the **Edit Problems** column. At the bottom, you can "Add blank problem template to end of homework set". If you wish for n problems to be drawn from the list in

`section2_4problems`, then add *n* blank slots. If you want to add more than 20, you will need to do so in stages.

Set the **Source File** for each of the blanks to be `group:section2_4problems`. Now set the options for the quiz and assign it to your students.

Importing and Exporting Assignments

WeBWorK allows for you to be efficient in managing your courses from term to term. You can package up details of homework sets and quizzes all at once and then move them to a new course.

Exporting an Assignment

At the end of the term, you may have put together several assignments and quizzes that you wish to keep for future courses, so that you do not have to build assignments from scratch in the future. To do this, you can **Export** assignments from the **Hmwk Sets Editor**. This creates a `.def` file containing the list of problems, how many attempts you chose, etc. If all of your problems came from public libraries and your header files were generic, then this will be enough. You can use the **File Manager** to download the `.def` file, and import it later into your new course. A `.def` file is just a plain text file; it would be harmless to open one and you might learn something by seeing its structure.

If the set used problem files that were local to your course (for example, if you modified a preexisting problem or wrote your own problems) then you must also copy these problem files. *And* you will need to copy the header (set and hardcopy) files that you were using. *And* you *may* need to copy the contents of your `macros` folder. You can do this efficiently by highlighting

- all of your problem and header files (`.pg`) in the **File Manager** (including any directories where you may have placed `.pg` files)

- all of the set definition files (`.def`) that you made

- the `macros` folder

and selecting **Make Archive**. You will probably need to ⌨Ctrl-click or Apple-click to select the three sources simultaneously. This creates a `.tgz` file compressing everything you highlighted. Download this `.tgz` file and store it.

Importing Assignments

If you already have a `.def` file in your **File Manager**, you can **Import** it using the **Hmwk Sets Editor**. At the moment that you import it, you can set an "earliest date" and all of the dates from the `.def` file will be shifted so that the earliest date corresponds to this. Otherwise you will need to use the **Hmwk Sets Editor** to adjust the dates directly.

If you are importing a set that you are bringing from an old course, then you need to go into your **File Manager** and upload the `.def` or `.tgz` file that is described in Exporting an Assignment. WeBWorK should unpack the `.tgz` file automatically when it is uploaded to the **File Manager**, and then delete the `.tgz` file.

Orientation Assignment

There is an orientation that is stored in the file `setOrientation.def`. You most likely need to import the orientation yourself, as described in Importing Assignments. This Orientation tries to address the most common problems that students run into using WeBWorK, and also make them aware of its more helpful features. I recommend that you assign it for some kind of credit.

Problems With Problems

WeBWorK problems are often coded with randomization, and sometimes the randomization gets out of hand and causes the problem to break down. Or the author of the problem may have just made a mistake. Because WeBWorK is open source, these issues are quick to fix. Or you may just see a problem that you want to modify for other reasons.

Modifying an Existing Problem

You may find a problem in one of the problem collections that almost suits your needs, but not quite. For example, there are problems that ask students to solve inequalities and express the solution using *both* interval notation and set-builder notation. You might want to use this problem, but remove the set-builder portion.

To modify a problem, add it to your problem set in the usual way. Once it is part of an assignment, in the **Hmwk Sets Editor**, click the numerical link under the column **Edit Problems** corresponding to the set in question. Find the problem that you wish to edit, and click **Edit it**. A new tab or window will open.

Now you have been thrown into the world of WeBWorK programming, but if you just want to make a simple modification, you should be able to make it without having to learn the PG language. Be aware that any modifications that you make will not change the library's version of the problem, but instead write a new problem just for you on your local directory.

You will probably need to make three changes.

1. You'll want to find the section of the displayed text that you would like to change. The displayed part of a problem falls between the commands `BEGIN_TEXT` and `END_TEXT`. Or if it is a newer problem, between `BEGIN_PGML` and `END_PGML`. Find the part that you want to delete/edit and do so.

2. If the problem is older and uses `BEGIN_TEXT` and `END_TEXT`, then answer blanks are produced in this same section with the command `ans_rule()`. You may need to remove or add these.

3. Scroll down to the bottom of the problem to see where the correct answer is compared with what a student puts into an answer blank. Usually, the command is something like `ANS($ans->cmp());`, where `$ans` is the variable storing the correct answer. You may need to remove, edit or add these commands too. Newer problems that use the `PGML` environment might declare the answer immediately after the answer blank instead of within an `ANS` command.

If the problem is newer and uses `BEGIN_PGML` and `END_PGML`, then the fix might actually be so intuitive that you don't need special instruction. Otherwise, ask for help.

After you have made your alterations, use the **Update** (for local problems) or **NewVersion** tab (for problems from a Library that you wish to make local) to save them. If making a new version, be sure to use the **Replace current problem** button as you save. Make sure the file is saved with a `.pg` extension. Your new and improved version of the problem has replaced the original in the assignment.

Writing Your Own Problems

If you would like to learn how to write your own problems, contact Chris, Alex, or Carl. You might get started by going to http://webwork.maa.org/wiki/Category:Authors. To examine code for a problem, find the problem in the **Problem Library** (or an assignment), and click to **Edit it**. You won't be editing the library's version—you'll be creating a new file on your local directory that can only be seen by you, not other instructors.

Theoretically, the sky is the limit as far as what can be programmed. Not only does the Perl basis of the PG language provide for excellent algorithmic programming, but you can incorporate Flash and Java applets if you get real crazy. Contact someone with experience if you are interested and they will point you in some right directions.

If A Problem Turns Out To Be Truly Bad

You can delete a problem from an assignment or choose to mark it correct for everyone. In the **Hmwk Sets Editor**, click the numerical link under the column **Edit Problems** corresponding to the set in question. Problem by problem, you can take either of these actions.

When You Receive a Student Email

Students will ask you for help with problems through WeBWorK's **Email instructor** button. These emails come with a link to the page from where the student sent the request for help. Click that link, and you'll be asked to log on to WeBWorK. You will be logged on as yourself, but (as you will see in the upper right corner of the screen) you will be acting as that student. This means that you

will be seeing things from their perspective, in addition to all the extra menus and commands that instructors have access to.

A useful tool to help diagnose their problem is to click the **Show Past Answers** button at the bottom of the problem they are having trouble with. This opens another tab or window where you can see *all* of the attempts they have made thus far. Most of the time, you can quickly diagnose their issue.

Write them back quickly and you will really impress them. However they need to check their actual email—WeBWorK itself will not alert them that they have a message from you. I recommend setting up your email to automatically tag these messages and filter them from your inbox. For filtering, all these messages have [WWfeedback] in the subject line. At regular intervals when you feel you have the time and are free from distractions, reply to your students.

Assessment

WeBWorK provides both formative and summative assessment tools.

Scoring

Use the **Scoring Tools** to compute student grades on assignments. You may do this as often as you wish, just be sure that you wait until the due date to copy scores into your regular grade book. Note that when you score a problem set, a .csv file is created that you can open with Excel. You may need to delete rows from this .csv file that correspond to instructors, administrators, and dropped students before you copy and paste into the grade book.

Statistics

The **Statistics** menu helps you analyze class performance as a whole on assignments and individual problems. If you are vigilant, you can monitor which problem numbers cause the most issues for students and go over these in class. This is an excellent formative assessment tool and you may develop a weekly habit of using it to choose a review topic for class.

Student Progress

View student progress with **Student Progress**. This shows you how deep into the assignment students are, individually or as a class. If you suspect too many haven't even started, you can crack the whip.

Editing Data: Change a Single Student's Scores, Attempts, or Due Dates

Sometimes you may wish to give students credit for a problem even though WeBWorK wouldn't. Or you might wish to add extra attempts to a problem for a student. Or you might want to give one particular student an extension.

In the **Hmwk Sets Editor**, click the numerical link under the column **Edit Assigned Users** corresponding to the set in question. Each student has a link to the left to **Edit data** for them.

Status is a number (0 to 1) describing their score on each problem. You can change that here. Be sure to click **Save Changes**.

If you change due dates or attempts for that student, you must *also check the box* next to the field. Again, be sure to click **Save Changes**.

'Gamification' of WeBWorK with Math Achievements

Some research and a lot of anecdotal evidence suggests that we can improve our students' homework completion rates by adding gaming elements to the assignment. Users of ALEKS will be familiar with its pie, which is an example of this.

Math Achievements

Geoff Goehle from Western Carolina university developed a WeBWorK module called Math Achievements to bring gamification to WeBWorK. If you enable it, then students earn math achievement points alongside the normal points they earn on assignments in two ways. Firstly, they can earn achievement points for every correct problem they answer in their assignments. Secondly, when certain achievements are met (like The Early Bird, where a student has 100% on an assignment within 24 hours of it opening) they get more points and a badge. Instructors can do what they will with these points—give some kind of extra credit or not. But as points accrue and tasks are met, students can "level up", eventually reaching "Level 10 Professor".

If you enable Achievements, there are two options: *without* items and *with* items. Achievement items are earned when a student levels up. They are one-time use items that the students can use to get an edge on their grade, while having a little fun. The items earned at each level (in default conditions) are

1. Lesser Rod of Revelation—Gives a student an automatic 50% on a single homework problem.

2. Potion of Forgetfulness—Resets the incorrect attempts counter to zero for a single problem.

3. Tunic of Extension—This allows a student to extend the due date of a currently open homework for 24 hours for full credit.

4. Cupcake of Enlargement—Doubles the weight of a single problem on a single homework set.

5. Box of Transmogrification—Allows a student to change the source file of a single problem to that of a different problem on the same homework set.

6. Greater Rod of Revelation—Gives a student an automatic 100% on a single homework problem.

7. Robe of Longevity—This allows a student to extend the due date of a currently open homework for 48 hours for full credit.

8. Cake of Enlargement—This doubles the weight of all of the problems on a currently open homework assignment. A student could end up having a 200% for their grade on one assignment if they use this.

9. Scroll of Resurrection—This reopens any homework set (even a closed one) for an additional 24 hours. The problem seeds are re-randomized, so the student won't be able to simply write down the answers while the set is closed, then resurrect it and enter them.

10. Greater Tome of Enlightenment—Gives a student an automatic 100% on every problem in a set. This is a reward for having been diligent with homework throughout the term and cannot be earned without a lot of achievement points from badges and answering questions.

Using Achievements without items

If you would like to use Math Achievements without items, the first thing to do is go into the **Course Configuration**, enable Math Achievements, and set the number of points that students earn for each problem[5]. This will add the **Achievements** tool to all users' tool panels, and the **Achievements Editor** to your instructor tools panel. In the **Achievements Editor**, import the default achievements (`default_achievements.axp`) and enable whichever ones you like. Achievements need to be 'assigned' to your students. If you get creative, you can also write your own achievements by using the **Edit Evaluator** link for an existing achievement.

Using Achievements with items

If you would like to use Math Achievements with items, the first thing to do is go into the **Course Configuration**, enable Math Achievements *and* Achievement Items, and set the number of points that students earn for each problem[6]. This will add the **Achievements** tool to all users' tool panels, and the **Achievements Editor** to your instructor tools panel. In the **Achievements Editor**, import the default achievements (`default_achievements_items.axp`) and enable whichever ones you like. Achievements need to be 'assigned' to your students. If you get creative, you can also write your own achievements by using the **Edit Evaluator** link for an existing achievement.

---

[5]The default setup uses 5 points per problem and is based on about 200 problems per term. You can set the points per problem proportionately.

[6]The default setup uses 5 points per problem and is based on about 200 problems per term. You can set the points per problem proportionately.

Challenge Problem Achievements

The default achievements that you can import are mostly ready to use immediately. The exception is the achievements that are classified as `04_challenge` achievements. Here the idea is to have award badges for completing especially challenging problems. There are two ways to implement these:

- Create a special problem set that has all of the challenge problems that you intend to assign for the term. If you make the name of this set be `Challenge`, then these challenge achievements will work immediately. You can have up to ten problems in it. For more, you would need to create more achievements by using the **Edit Evaluator** link on the 10th challenge achievement and writing an 11th, etc.

- Locate the problems in your various problem sets that you would classify as challenge problems, and take note of the names of the sets they are inand what problem number they are within that set. Then use your **Achievements Editor** to **Edit Evaluator** for the challenge problem achievements. Where you see code like

  ```
  my %validproblems = (
  'Challenge' => {
      '1' => 1, }
  );
  ```

  change the code to use the name of the homework set with the challenge problem and the problem number from that set. For example, if the the challenge problem is number 13 from the set `Section2_4`, you might end up with

  ```
  my %validproblems = (
  'Section2_4' => {
      '13' => 1, }
  );
  ```

Of course a third option is to just disable these challenge badges. And there is no harm in just leaving them as they are if you do not feel any problems that you have assigned are worthy of being called challenge problems. However, if you have the time and have some challenging problems, it adds a little more fun for your students.

If you have made edits like this to the challenge achievements, you would want to consider the implications for exporting these changes to a new course in a subsequent term. Specifically, you would use the **Achievements Editor** to **Export** all of your chosen achievements to a `.axp` file. Then you would want to include the `achievements\` folder during the export process.

Assigning Achievements to lately-added students

There may be times when you add students to the class after it has begun; once you have added the students to the class, you will need to assign the Achievements to them as a separate action.

Go to the **Achievement Editor**, and you should see that not all of the users in the class have been assigned the Achievements. Click on **Assign** and then **Select all**; be sure to keep the drop down box as `preserve existing data` so as not to erase the existing data of your currently-enrolled students. Once you have clicked **Take Action!**, you should see that all of your users have been assigned all of the achievements.

Show Me Another

Commercial online homework platforms often have a feature where a student can get a step by step solution for the problem they are working on, at the cost of having the problem that they were assigned re-randomized. WeBWorK's Show Me Another Feature is like this, except that the version of the problem that was originally assigned to the student remains their assigned version, and they can see solutions to the *newly* seeded problem. It is worth noting that if WeBWorK is unable to generate a new version of the problem that is different from the version that is actually assigned, then it will tell the student this, and that this feature is unavailable for this problem.

There are several checks in place to allow you to customize how this feature is used. First of all, in the **Course Configuration** you decide what elements of the newly seeded problem the student will have access to. There are four options, any subset of which can be enabled.

- Walk-through solutions: if the newly seeded version has a walkthrough solution, then the student will be able to see it. This is the option that makes SMA a parallel to MyMathLab's Show Me How button.

- Answers: this is redundant if you have enabled the walkthrough solutions and there is a walk-through solution. But otherwise, it let's students see what answer was expected for the newly seeded version.

- Hints: some WeBWorK problems are coded with hints that are revealed depending on a global setting by the instructor and how many attempts the student has used. With this checked, students have access to the newly seeded version's hints regardless of other settings.

- Check answers: this gives the student the ability to enter an answer for the newly seeded version and have WeBWorK check if it is correct. This would be most useful if the walk-through solutions and answers have not been enabled.

Personally, I tend to always offer SMA with the Check answers option turned on, and all other elements turned off. This gives motivated students opportunity for extra practice. There's nothing wrong with turning on the other options, but it's just not my style. I *do* turn on all the other options if I am creating a WeBWorK course whose purpose is self-study instead of homework.

The other thing to set in **Course Configuration** is the number of times a student may use **Show Me Another** for each problem. Maybe you wouldn't care to allow your student to use this button over and over again on any one problem forever. Personally, I set this to -1, which is code for "there is no limit".

Lastly, you very well might like this feature to be enabled for some problems, but not all. For this reason, in the **Set Details** page for a problem set (accessible through the **Hmwk Sets Editor**, and then clicking on the number of problems within the set), you can set the threshold of attempts that a student must use before the button becomes available to them. For example, it is set to 2, then they must attempt the problem twice before the button is available. If you want the button to be immediately available, set the threshold to 0. If you want the button to never never be available for that problem, then convention is to use −1.

Important: If you are importing one of the PCC default homework sets, then every problem's threshold number will be −1. If you want them to be 0, or 2, or whatever, it would be extremely tedious to change the threshold manually for each problem. So *before* you import the problem set(s), go to the **File Manager**, navigate up one folder, and edit the file `course.conf`. Find a blank line anywhere near the bottom and paste the line

$$\texttt{\$problemDefaults\{showMeAnother\}2;=}$$

or of course use some number other than 2. Save, and *then* import your problem sets. They will all have their SMA threshold set to whatever setting you made.

Good Ideas to Keep Everything Running Smoothly

- Respond as soon as you can to students' requests for assistance. A fast response rate really improves students' attitudes about using WeBWorK. Students cite this fast response rate as one of the top two reasons for liking WeBWorK. (The other is the immediate feedback from WeBWorK itself.)

- Test the problems that you assign by assigning the homework sets to yourself, and running through all the problems. Many of the problems are freshly programmed and there are bound to be some mistakes. Right now the easiest way to handle this is to use the **Email Instructor** button from the offending problem to send yourself a message. Then forward that email to Alex, Carl, or Chris.

- While you do the problems yourself, make note of any special 'computer language' that is necessary for completing the set. For example, that you enter π as `pi`. Write all this down and include it in the set header. Mention these items in class too.

- Consider giving unlimited attempts to problems. If the underlying philosophy is that homework is a learning device and not a testing mechanism, then the good that comes from unlimited attempts outweighs the bad.

  - Students are encouraged by the ability to keep trying to solve a problem until they get it right.

  - There will be some problems where students will swear up and down that they are correct, yet WeBWorK marks them incorrect because they 'do not know the computer language'. (Sometimes this has merit, but much more often they simply have wrong answers.) They

become really upset *if they use up all of their attempts* while completely confident that they are correct. A negative attitude is more than just a pain for you – it could reduce the effectiveness of WeBWorK as a learning tool.

  – Suppose that a poorly programmed question escapes your notice during your preview of the assignment. Suppose that you make the call to alter the problem to fix the issue, as is outlined in a section above. Suppose that students have *limited* attempts. By the time you make your changes, several students may have used up all their attempts, and you would need to deal with that.

  – I find that some proud/too-polite students will not ask for help until they have tried many (>10) attempts. With limited attempts, these students might just move on after using up their attempts, and never ask for help.

There will rarely be an opportunity for exhaustive guessing to lead to correct answers. When a problem does lend itself to exhaustive guessing (e.g. a matching problem), of course that is a good time to limit attempts.

- Use the **Statistics** menu to identify which problems are generally causing trouble for your students that you could go over in class.

- Use an 'attempt threshold' to encourage an email request for help. For example, you might say "Email me for help if you are still stuck after five attempts."

- Make sure students are aware of the tools they have:

  – The **Email Instructor** button
  – The feedback messages (which students often overlook)
  – Printing hard copies
  – PCC computer access
  – Viewing answers after the answer date

The Orientation assignment tries to make them aware of most of these, but redundancy (like mentioning these things every now and then) is helpful.

- Give your students a short guide similar to this one, but for students. (I've made one that you are welcome to use or edit.)

- If you are using questions that come from a textbook, assign even numbered problems (problems without answers in the back of the book.) Obviously, you don't want a slacker student to just enter the answers from the back. More importantly, the reasoning behind assigning odd problems is one-upped with WeBWorK. The purpose of assigning odd problems is so that students can see for themselves when they are incorrect. WeBWorK can do this *without* simultaneously giving away what *is* correct.

- If you are using problems that have randomization, encourage students to work together. They will not be simply able to give each other answers; they will need to communicate process to each other. This kind of collaborative learning is good for all involved, and the randomization serves as a countermeasure to outright cheating.