

Code Snippets

**Justin A. Roman, Tanisha Rankin,
Grae Cullen, and Paul Wolfgang
Fall Semester 2008**

**CIS 4339 – Project in Computer Science
Temple University Computer Science
Professor Paul Wolfgang**

Table of Contents

- Intro 3
- Development Tools 4
 - Backend:..... 4
 - Frontend: 5
 - Versions/Architectures 5
- Technical Matters 6
 - Database Diagram 6
 - Sample Problem XML..... 7
 - Log In/Log Out 8
 - Add Category..... 9
 - Add/Edit Problem 10
 - Add/Remove a Teacher/Student..... 11
 - Running Student’s Code..... 12
 - View Student’s Completed Work 13
 - Security 14
- Using the Application 15
 - Student and Teacher: 15
 - Teacher: 17
 - Student: 22
- Project Future 28
- Known Issues..... 29

Intro

The purpose of this project was to develop a website which will primarily be used as a teaching aid for computer science teachers/professors teaching the C language. The secondary objective of this website is to allow students, or any other individual, to freely practice modern coding techniques. Professors will be allowed to assign pre-defined problems which will be listed categorically based on level of difficulty. Completed code will be submitted by the student where the server will save, compile, run, and check the method. Pre-defined test cases will determine the validity of the student's code. User logins will allow the professor to check each student's individual progress, which is updated after each properly completed problem.

Development Tools

Backend:

Java

Java is the backbone programming language of our project. We decided to use it because of its ease of use and ease of integration with JSP and MySQL. The main reason for using an object-oriented programming language such as Java is that each problem can be an instance of a Java class. We created a Problem Java class to represent the many different problems found on our website. The attributes of the problem being viewed are loaded in from the database.

JSP

JSP was used as our server side scripting language because it provides a simplified way to dynamically generate web pages. On the server, a JSP engine interprets JSP tags and scriptlets, generates content (JavaBeans, JDBC, or include files), and sends the results back in the form of an HTML page to the browser. This ensures that our pages can be hosted on any HTML-based web browser.

JSTL

JSTL is a collection of JSP tags that provide standard commands used to control how data is retrieved and displayed. We used JSTL to access the database, and to set page and session variables.

MySQL

MySQL is the database software package that we use in many different ways throughout our application. When deciding which database service to use we came to a decision rather quickly. MySQL is a free software package that works seamlessly with all three of the abovementioned technologies. SQL commands such as insert, update, and delete can all be easily written and manipulated in Java, JSP, and JSTL.

Apache Tomcat

Apache Tomcat was chosen as our web server because Tomcat implements the Java Servlet and the JavaServer Pages, and can provide a Java HTTP web server environment for Java code to run.

GCC

GCC, or GNU Compiler Collection, is the compiler used behind the scenes when a program is to be run by a user or teacher. We needed a compiler to assemble the C code we pass it in a Unix environment.

XML

XML is used when storing all of the problem information in the database. We decided to use XML rather than database columns because 1) there are a dynamic number of testpoints for each problem and 2) it is easy for the Java code to parse through XML.

Sandbox

Sandbox is a program written by Professor Paul Wolfgang with the purpose of providing security to our website. Sandbox is used every time a program is run on the server side. Sandbox limits the user by disallowing him/her access to our source files. Sandbox was written in C++ because of the ease of string manipulation.

Frontend:

HTML/CSS

A lot of the programming, besides the JSP and Java code, was done in HTML since this is a web application. CSS was used throughout our website to help with the navigation bars, the body text, and some other aspects of the site.

JavaScript

JavaScript is used mostly for client side form validation, and to enhance page convenience. JavaScript was used for returning to the editProblem.jsp and addProblem.jsp pages after a submission. It is also used to load initial values into forms, as well as setting the focus of a page to a specific field. Additionally, as an added convenience for the user, JavaScript was used to give the user the ability to use tabs within the text area on the problem.jsp, addProblem.jsp, and editProblem.jsp pages.

Versions/Architectures

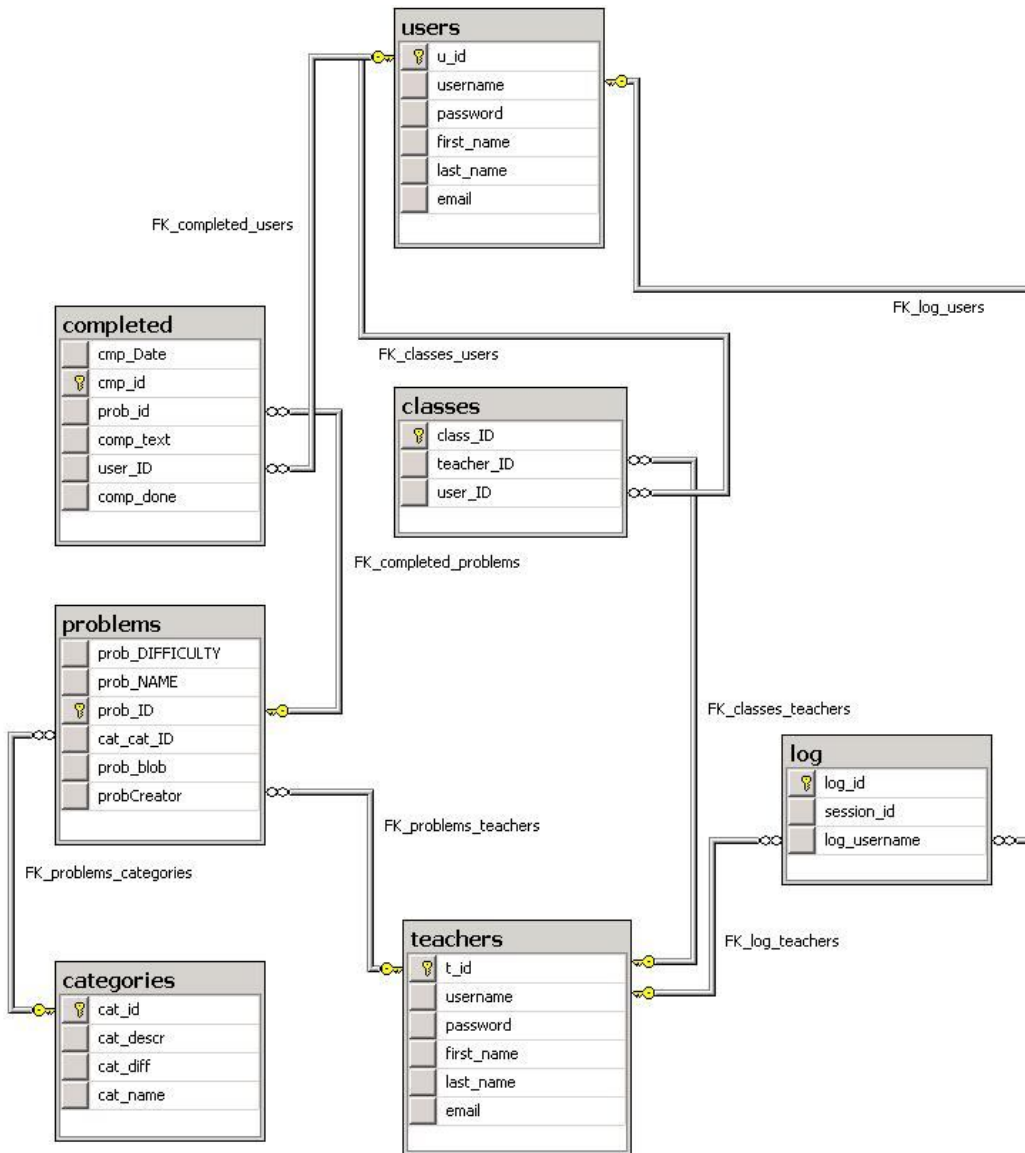
Apache Tomcat – 6.0.18

JVM by Sun Microsystems Inc. – 1.6.0_07-b06

Linux (AMD64) – 2.6.25.9-40.fc8

Technical Matters

Database Diagram



This is the list of tables that are used to store all pertinent information concerning our application, as well as the keys that connect each table. All user information (name, email, and which user is logged into the system), problem and category information, class registration, as well as keep track of the student's progress. The XML page for the problem is stored in the problem table as a BLOB (Binary Large Object).

Sample Problem XML

```
<problem>
  <listing>
    <title>Add Doubles</title>
    <funcName>addDoubles</funcName>
    <funcCall>addDoubles(arg1, arg2)</funcCall>
    <description>Write a function that will return the sum of the two double arguments.</description>
    <prototype>int addDoubles(int numOne, int numTwo);</prototype>
    <returnType>int</returnType>
    <numArgs>2</numArgs>
    <args>
      <type>int</type>
      <type>int</type>
    </args>
    <starttext><![CDATA[
int addDoubles(int numOne, int numTwo)
{

}]]></starttext>
    <completed><![CDATA[#include<stdio.h>

int addDoubles(int numOne, int numTwo)
{
    return numOne + numTwo;
}]]></completed>
  </listing>
  <testpoints>
    <test>
      <inputVals>
        <input>-2</input>
        <input>-6</input>
      </inputVals>
      <expected>-8</expected>
    </test>
    <test>
      <inputVals>
        <input>0</input>
        <input>3</input>
      </inputVals>
      <expected>3</expected>
    </test>
  </testpoints>
</problem>
```

The above is an example XML object describing all the fields required for a problem. Each problem XML object is broken up into two parts, namely the “listing” section and the “testpoints” section. The “listing” piece, as shown above, contains all the information that describes the problem including the starting text and the completed answer.

The “testpoints” section contains the multiple tests created by the teacher. The inputs are taken from what the professor enters and the expected return value for each test is derived from running the teacher’s program with the given input values.

This XML write-up is zipped and saved in the database in addProbProcess.jsp. It is then later unzipped and the information is extracted using an XML reader when the specific problem is being viewed in problem.jsp.

Log In/Log Out

Log In

When a user attempts to log in to our website the first thing we do is check the submitted user name and password against our database. If the user selects the radio button for student, we check the information against the “users” table and if the radio button for teacher is selected, we check the information against the “teachers” table. If `loginProcess.jsp` determines that the information is invalid, we redirect the user to `login.jsp` with an error message describing the issue that arose during the login attempt. If the submitted criteria match our database, the user is redirected to `categories.jsp` but before that, there are a couple of actions performed. First, we check to see if the user is in the “log” table of our database. If the user’s user name appears then this means he/she has a session elsewhere (whether or not this is on the same computer). After removing any occurrences of the user name in the “log” table, we insert into the table the newly created session ID and the user’s user name. This allows for only one login session per user at a time since every page checks the user’s current session ID against the one listed in the “log” table of the database. During the second action performed we create all the necessary session attributes in order for the student/teacher to successfully navigate through our site. These attributes include a user ID, first name, last name, account type (student or teacher), and a maximum inactive interval.

Log Out

Logging out of our website invalidates any session variables created during the login process. The page, `logout.jsp`, also deletes the row that contains the current session ID from the “log” table. Logging out makes sure that no one can use our website as the previous user since every page checks the current session ID against the one listed in the “log” table of the database.

Add Category

Only a teacher has the ability to add a new category to the database. The category name, description, and difficulty are all required when doing this. The aforementioned information is taken from the submitted addCategory.jsp page and added into the categories table of the database. This event takes place in the addCatProcess.jsp page. A simple SQL insert statement is performed to add this information to the database.

Add/Edit Problem

From 'C' to Problem Object

The first step in storing a problem is to create the Problem Object for a given problem from the user input. This is done with the ProblemParser object. The ProblemParser object takes the problem name, problem description, the completed function, and a String array, 1 or more dimensions, of test points, and takes that data and constructs a Problem object. The Problem object is a Java representation of the problem. From it you can get data such as types of the arguments, or return type. The parser will extract that information from the teacher code. However, the "correct" answers are obtained by actually running the code. In order to run code, we create a ProgramBean object, set the ProgramBean's problem as the current problem, and call the teacherTest function. From here there are a number of possible paths for the code to run.

Results

The result is either submission success, or one of five different failures. The first failure is in the case of a certain test point did not compile. This can be caused by invalid c-code, or invalid syntax in the test points. Anything that would fail to compile on GCC, gets sent back with a message about syntax and the GCC compile error/warnings message. The second failure is a failure of the parser, that is, the parse is unable to parse the function submitted. The only way this can be caused is if the function has unsupported options, such as return type of a pointer (*), other than character. Although it works with C code, if you type in complete junk it will not handle it very gracefully. However, in the future it could throw the InvalidFunctionException in more situations. These could all be handled in the current JSP code. The third type of submission failure is the InvalidTestPointsException, which is caused by putting the wrong number of arguments in for a function, compared to the number of arguments you have in the test points. The fourth type of submission failure is if there is a Java exception during the building and running of the GCC compiler process. This has yet to happen. Any generic Exceptions that might happen beyond these planned paths will be caught by our error.jsp page. Submission success, on the other hand, adds the newly created problem to the database. If the problem, cannot be added to the database, that is the fifth submission failure. This also, has yet to happen.

From Problem object to XML

In order to add a Problem object to the database, it is converted into XML. Not only is the actual text of the function added, along with the description and test points, all the parsed data such as return type and number of arguments is added to the XML string also. This XML string is then compressed and added to the database using a BlobBean object.

Edit: Full Circle back from XML

The process for editing a problem is generally the same, with one key difference. The original data used for creating the Problem object was read from the XML in the problem.jsp page. The process is nearly the same, with one difference being that the database is being updated with a SQL update statement rather than a SQL insertion.

Add/Remove a Teacher/Student

Since this system will be used as a teaching aide, we will allow the students to share all of his/her completed programs with a teacher. The student user must have the teacher's email address in order to add the teacher. The student will go to the myTeachers.jsp page that will list, in table format, the name, email address, a button to view a teacher's problems, and a button to remove a teacher. This table includes any teacher the student is currently sharing their work with. If the student is not currently sharing their work, then a message will be displayed. The bottom of myTeachers.jsp has a form that has a text box for the student to provide the teacher's email address. This form will submit to the share.jsp page, a processing page that will access the database and attempt to insert the student's ID and the teacher's ID to the "classes" table. If the student has already entered the teacher's email, the student will be forwarded back to the myTeachers.jsp page and an error message will be displayed. However, upon successful insertion, the student will still return to the myTeachers.jsp page and the table will be updated to include the new addition. If the student decides to stop showing their work to a teacher, they simply click on the remove button on the table. This button will submit to the removeClassID.jsp page, a processing page that will delete from the classes table the row containing the specified teacher and student. A teacher user has a page similar to the student's myTeachers.jsp page called myStudents.jsp. This teacher's page has all of the same functionality except the teacher does not have the capability to add a student to his/her class.

Running Student's Code

Preparation

The `problem.jsp` page either loads the student's last attempt from the database or puts the start text into the submission box if there is no previously made attempt saved in the database. The student can then change the code as needed.

Submission

When the student hits the "Submit" button, a number of things happen. The `results.jsp` page is included where the student's results are checked against the expected values. If the student has not submitted the problem before, a SQL insert statement is performed where the submitted text is saved along with completion date and a Boolean value of whether or not the submitted answer is correct. On the other hand, if the student has submitted before and they are simply resubmitting his/her newly edited solution, a SQL update statement is performed updating the fields described earlier.

Hitting the submit button also causes `results.jsp` to be included in the `problem.jsp` page. This page actually displays the results that the student gets for their problem. Additionally, it updates the database as to whether the student has submitted a correct answer.

Testing the Student Submission

In order to actually test the submission, the `results.jsp` page gets the number of test points from the already initialized `ProgramBean`, and then runs `runProgram` function for each of the test points. Each `runProgram` call will return true if the entered C code compiled and ran. If the code did not compile and run properly, we display the message in the results box, and discontinue testing the rest of the test points. However, if the code does compile and run, we change the `getProgramOutput()` call with the `getExpected()` call. If the two returned strings are equal, we print out a green block in the table. If the two strings do not match, meaning the answer is incorrect, we print out a red background in the table square indicating success or failure.

The Actual Running of the Program

The `ProgramBean` runs the program with the following steps. First, the `runProgram` method makes a new directory in which to put the C file. Then the actual C file is written. The user entered function is printed to the file. Next the rest of the C file is produced by the `Program` object's `cFile` method, and written to the file. The C file is then compiled into an executable. We then use the sandbox program to run the executable, in a "sandbox" which does not allow the executable access to any system files. The sandbox program actually changes root, and runs the program as a non-privileged user in that new root directory. That sandbox program will return zero if the program ran successfully. If the sandbox program returned successfully, the output from that program is simply returned from the `runProgram` Java method to the caller. If the return value of sandbox is nonzero, that means the program timed out. In the case of a time out the `runProgram` function returns a String "Program Timeout." The `runProgram` method will also return the compiler error messages, if the program does not compile. The method may also catch an Exception, and return a string representation of it.

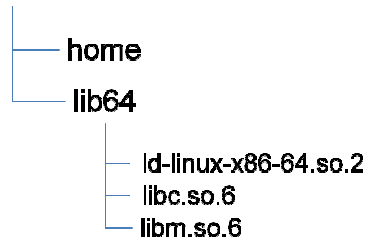
View Student's Completed Work

A teacher has the ability to view all of a student's completed problems by going to the `myStudents.jsp` page. Once a student has registered for a teacher's class, the `myStudents.jsp` page will provide that student's information in table format. For each student, it will list their name, email address, a button to view the student's problems, and a button to remove the student. By clicking on the view problem button the teacher will be redirected to `studentReport.jsp`. This page will list the students correctly completed problems. The table is generated by accessing the database and getting the information from the completed table. This includes completion date, category name, problem name (with a link to the problem), and the problem text. The teacher can view a student's answer to a problem in a read-only text area. If a teacher deletes a problem that a student has already completed, this table will be updated to reflect the change.

Security

Sandbox

sandbox_base



The base sandbox is in the directory `sandbox_base`. It contains the above directory structure. The `home` sub-directory is empty.

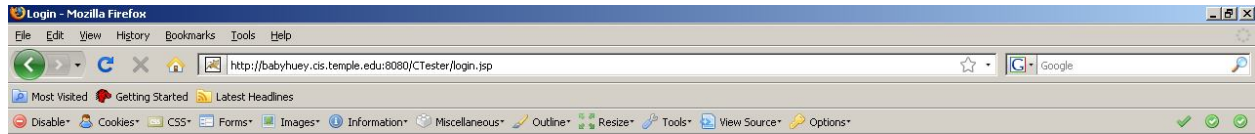
Sandbox performs the following steps:

1. Removes any directory of the name `sandbox_user-name`
2. Creates a directory `sandbox_user-name`
3. Copies `sandbox_base` to `sandbox_user-name`
4. Creates a directory `sandbox_user-name/home/user-name`
5. Copies the user's program to `sandbox_user-name/home/user-name`
6. Changes the owner of `sandbox_user-name/home/user-name` to `user-name`
7. Changes the root to `sandbox_user-name`
8. Performs a fork.
9. In the child process:
 - a. sets the user-id to the user-name's user-id
 - b. limits the process to a single process (no children)
 - c. arms a 5-second time out
 - d. executes the program
10. In the parent process:
 - a. Wait for the child process to terminate
 - b. Return the exit status of the child process.

Using the Application

Student and Teacher:

Log In



Code Snippets - Login Page

To enter our site please enter your login id and password below:

Please remember **both** fields are case sensitive!

Login ID

Password

Account Type Teacher Student

If you are not registered please do so [here!](#)

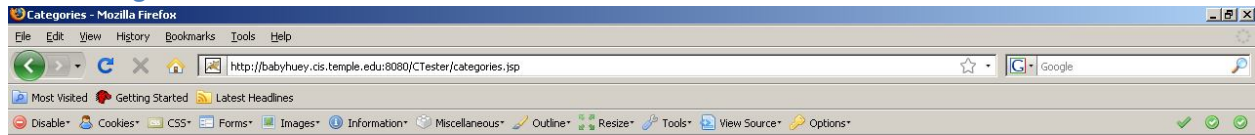
Done

Logging into our website is very simple. Click the submit button after entering your registered login name, your password, and selecting the radio button that describes what type of account you will be logging in as.

Log Out

Logging out of our website is just as simple, if not simpler. Every page contains a navigation bar along the top and bottom with the rightmost link being labeled "Logout." Clicking this link will log you out of the system and redirect you to login.jsp.

View Categories and the List of Problems



Basic Arithmetic (Difficulty: 1)

- [Add Doubles \(Diff: 1\)](#)
- [C to F \(Diff: 1\)](#)
- [Triangle Area \(Diff: 1\)](#)
- [Modulus \(Diff: 2\)](#)
- [More...](#)

Loops (Difficulty: 2)

- [Print Multiples \(Diff: 2\)](#)
- [More...](#)

Strings / char * (Difficulty: 3)

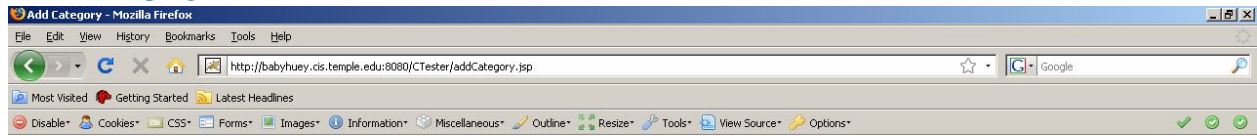
- [Dogs and Cats \(Diff: 3\)](#)
- [More...](#)



After a user is logged into our system, they will be redirected to the category page. This page is dynamically loaded with information from the categories and problems table. The categories are displayed in order of difficulty and then alphabetically. The number of categories displayed per row is pre-defined as four, and each category will list the first four problems ordered by problem difficulty. If the number of problems for a specified category is less than four, then just the available problems will be listed. Each problem name is a link to the problem.jsp page. The last link for each category is titled "More..." and this link will take the user to the displayCategory.jsp page. This page will display all problems for the specified category ordered by problem difficulty and then by problem name.

Teacher:

Add a Category



Add a Category

Category Name: (ex. Basic Arithmetic)
Category Description: (ex. Basic Math Problems)
Category Difficulty:



A teacher has the ability to add a category on this page. Adding a category requires that the teacher enters a category name, description, and difficulty for the soon to be created category.

Add a Problem

The screenshot shows a web browser window titled "Add Problem - Mozilla Firefox" with the URL "http://babyhuey.cs.temple.edu:8080/CTester/addProblem.jsp". The page has a navigation bar with links: Categories, My Students, My Problems, Add Problem, Add Category, Preferences, and Logout. The main content area is titled "Add a problem" and contains the following form fields:

- Problem Name:** (ex. Add Doubles)
- Problem Description:** (ex. Write a function that will add two double arguments, and return a double that is the sum of the values.)
- Problem Difficulty:** 1: Easiest
- Problem Category:** Basic Arithmetic: Difficulty(1)
- Complete Function:** (Must have Includes)

```
#include<stdio.h>

int sampleProblem(int sampleArgument)
{
    int correctAnswer;
    correctAnswer = sampleArgument + 1;
    return correctAnswer;
}
```
- Number of Parameters Your Function Will be Fed:** 1
- Set Number of Parameters!** button

In order to add a problem to the website, please follow these steps:

1. Enter a unique descriptive name.
2. Enter a problem description, which should tell the student exactly what the student must do in order to complete the problem.
3. Select a difficulty level for the problem. This level will be used in the sorting of problem. The easier problems are listed first.
4. Select a category in which to place the problem. You may add new categories elsewhere on the page.
5. The complete function must be the correct code for the problem described in the problem description. The student's answers will be matched against the answer this function produces. Even if the student is right, the website will assume the answers your code produces will be correct.
6. Indicate the number of arguments your function requires. This must match the number of arguments in the "Complete Function" section.
7. After you indicate the number of arguments, click the "Set Number of Parameters Button."
8. You will then be able to set the number of test points to use.
9. The site will warn you with a popup that changing the number of test points will remove and test points you have already saved. Click "Ok" to continue.

Complete Function:
(Must have Includes)

```
} return correctAnswer;
```

Number of Parameters Your Function Will be Fed:

Number of Test Points You Will Provide for the Problem:

Test Point 1:	<input type="text" value="test1arg1"/>
Test Point 2:	<input type="text" value="test2arg1"/>
Test Point 3:	<input type="text" value="test3arg1"/>

Categories
My Students
My Problems
Add Problem
Add Category
Preferences
Logout

Done

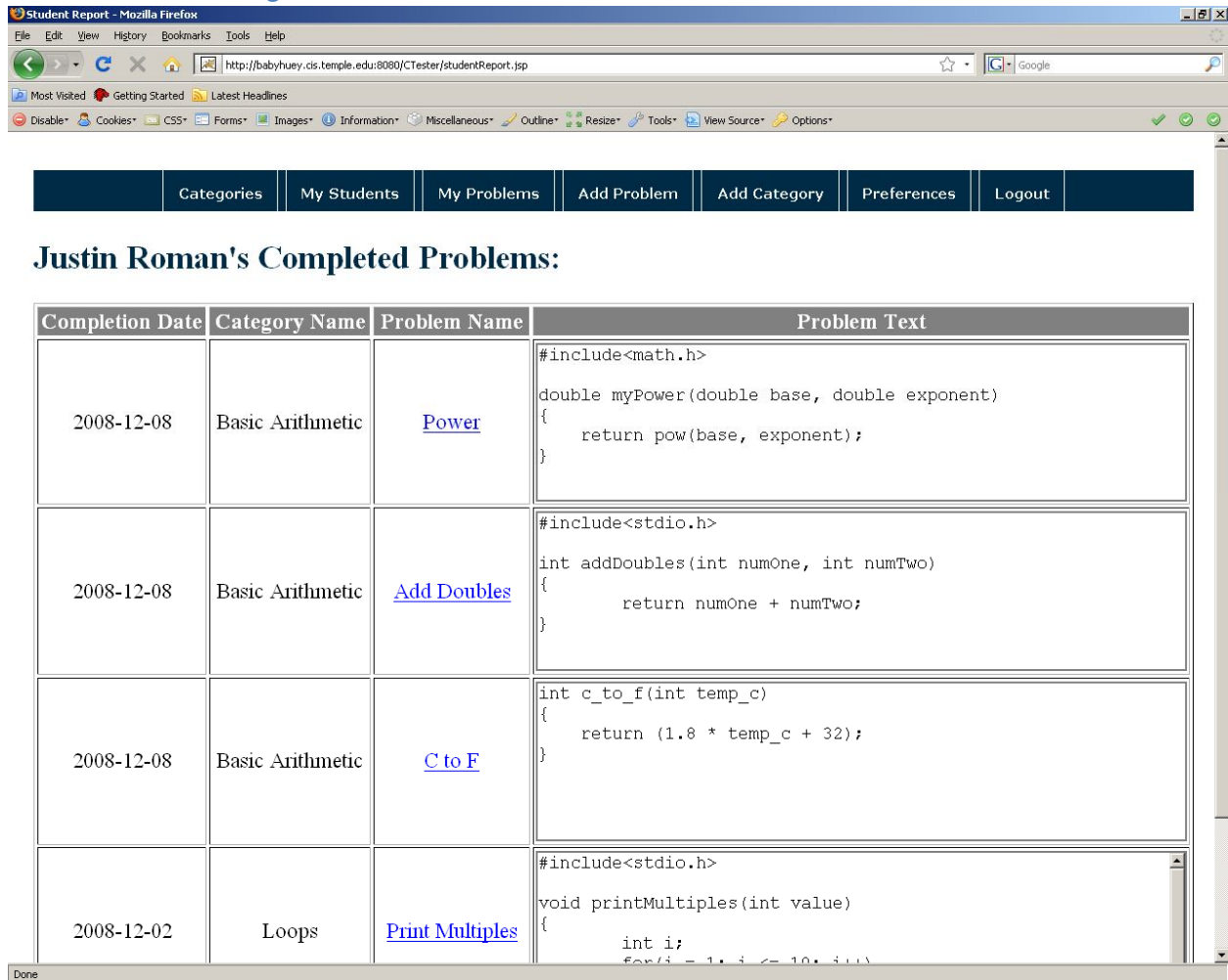
10. You are now about to type in test points.
11. Each test point is typed on a new line. If you have more than one argument, each argument is placed in a box in a particular row.
12. Click "Submit Problem!" when you are finished.

Assuming everything is correct, you will get a submission success. If not, you will receive a submission failed message and a reason as to why it failed. Click the back button, or "Redo Submission" button, and try to correct whatever mistake was made.

Edit a Problem

The process for editing a problem is nearly the same as adding a problem. There are two key additional things when editing a problem. First, if you change a problem, even in a superficial way, the student is not considered to have completed that problem. The student has to resubmit the problem, to see if its answers match the answers of the newly edited problem. Additionally, as the program is written, the test points are lost each time you edit a problem. So you will have to retype the test points each time you edit a problem. Another warning, avoid directly typing in problem ID number in the address bar, like "editProblem.jsp?problemID=95". This can cause a bug, you should first view the problem you want to edit, and then click the edit button. Skipping directly to the editProblem.jsp page can cause problems.

View a Student's Progress and Submitted Answers



Student Report - Mozilla Firefox

http://babyhuey.cs.temple.edu:8080/CTester/studentReport.jsp

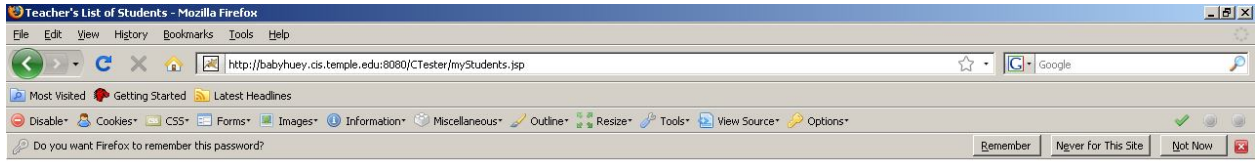
Categories | My Students | My Problems | Add Problem | Add Category | Preferences | Logout

Justin Roman's Completed Problems:

Completion Date	Category Name	Problem Name	Problem Text
2008-12-08	Basic Arithmetic	Power	<pre>#include<math.h> double myPower(double base, double exponent) { return pow(base, exponent); }</pre>
2008-12-08	Basic Arithmetic	Add Doubles	<pre>#include<stdio.h> int addDoubles(int numOne, int numTwo) { return numOne + numTwo; }</pre>
2008-12-08	Basic Arithmetic	C to F	<pre>int c_to_f(int temp_c) { return (1.8 * temp_c + 32); }</pre>
2008-12-02	Loops	Print Multiples	<pre>#include<stdio.h> void printMultiples(int value) { int i; for(i = 1; i <= 10; i++)</pre>

A teacher has the ability to view the progress of the students registered for his/her class. The page, `studentsReport.jsp`, lists the students correctly completed problems. The table includes completion date, category name, problem name (with a link to the problem), and the problem text. The "problem text" column includes the student's correct answer in a read-only textarea allowing the teacher to view how the student went about answering the problem.

View a Class/Remove a Student from Your Class



Paul Wolfgang: below is the list of students registered for your class.

Name	Email Address	Completed Problems	Delete
Cullen, Grae	tua57044@temple.edu	<input type="button" value="See Problems"/>	<input type="button" value="Remove Student"/>
Rankin, Tanisha	nikkrank@gmail.com	<input type="button" value="See Problems"/>	<input type="button" value="Remove Student"/>
Roman, Justin	jroman00@gmail.com	<input type="button" value="See Problems"/>	<input type="button" value="Remove Student"/>
Student, A	student@babyhuey.cis.temple.edu	<input type="button" value="See Problems"/>	<input type="button" value="Remove Student"/>

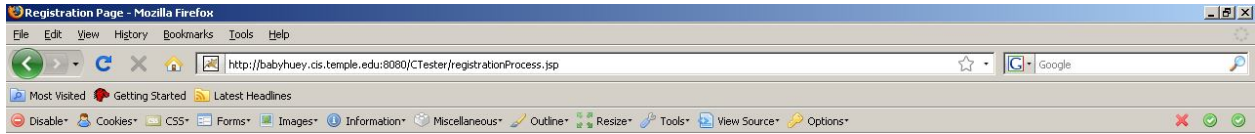


Done

The myStudents.jsp page allows the logged in teacher to view a list of his/her registered students along with email address. On this page, the teacher can click the button labeled "See Problems" which brings the teacher to the page previously described. The "Remove Student" button removes the student from this list. You can then no longer view the student's completed problems.

Student:

Register an Account



Student Registration Page

Please fill out the following form to join our site:

Please remember **both** login id and password fields are case sensitive!

A required field is missing	
Username	<input type="text"/>
Password	<input type="password"/>
Re-enter Password	<input type="password"/>
First Name	<input type="text"/>
Last Name	<input type="text"/>
Email Address	<input type="text"/>
Re-enter Email Address	<input type="text"/>
<input type="button" value="Submit"/> <input type="button" value="Reset"/>	

To register as a Teacher, contact administration.

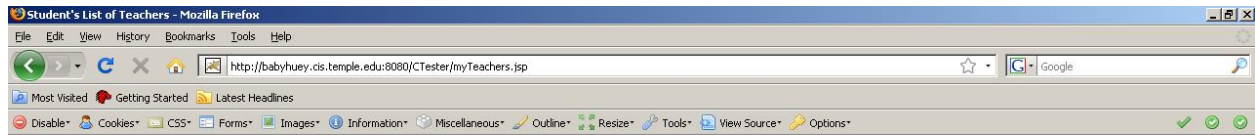
Go Back to [Login](#) Page.



The registration.jsp and registrationProcess.jsp pages allow students to register accounts. The student must enter all of the requested information with the two password fields being equal to each other and the two email address fields being equal to each other. If a field is empty, passwords or emails do not match, or the username requested is already registered the user will be redirected to this page with an error message along the top. In this screenshot, the error message says "A required field is missing."

NOTE: Teachers who wish to register must contact an administrator.

Enter/Exit a Class



You are currently sharing your completed problems with:

Teacher Name	Teacher Email	View Problems	Remove Teacher?
Cullen, Grae	grae.cullen@verizon.net	View Problems	Remove
Rankin, Tanisha	nikkrank@gmail.com	View Problems	Remove
Roman, Justin	jroman00@gmail.com	View Problems	Remove
Wolfgang, Paul	wolfgang@temple.edu	View Problems	Remove

Sharing your work with a teacher will make your completed problems visible to that teacher. :

Teacher's Email Address:	<input type="text"/>
	<input type="button" value="Add Teacher"/>

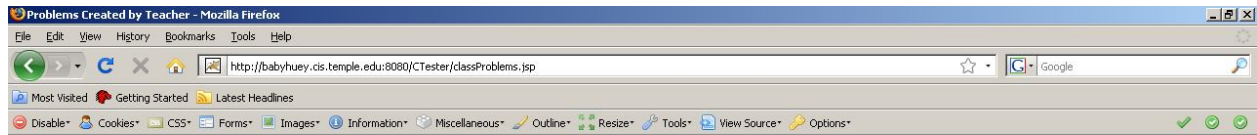


Done

The myTeachers.jsp page allows the logged in student to view a list of his/her registered teachers along with email address. On this page, the student can click the button labeled "View Problems" which brings the student to the classProblems.jsp page. The "Remove" button unregisters the student from the respective teacher's class. By doing this, that teacher can no longer view your correctly completed problems.

This page also allows you to enter a teacher's email address with whom you want to share your completed problems. The email provided must belong to a registered teacher otherwise an error message will be returned. By submitting a valid teacher email address, you are essentially joining his/her class.

View Class Problems



Paul Wolfgang's Created Problems:

Category Name	Problem Name	View Problem?	Completed?
Basic Arithmetic	C to F	View Problem	Green
Strings / char *	Dogs and Cats	View Problem	Red
Basic Arithmetic	Triangle Area	View Problem	Red



Done

This page, classProblems.jsp, can be accessed by clicking the “View Problems” link previously mentioned or by clicking on the teacher’s name found at the bottom of problem.jsp. The table includes four columns, namely category name, problem name, view problem, and completed. The “View Problem” button points the user to the problem.jsp page with the specified problem. The “Completed” column shows green if the student successfully completed the respective problem or shows red if the student either did not start the problem or did not complete it successfully.

NOTE: A teacher can view another teacher’s problem by coming to this page. He/she can do so by clicking on the teacher’s name found at the bottom of problem.jsp. The “Completed” column will not be displayed for teachers.

Compile, Run, Test, and Save Problems

Problem Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://babyhuey.cs.temple.edu/CTester/problemProcess.jsp

Categories My Teachers Preferences Logout

Basic Arithmetic → Add Doubles

Description:

Write a function that will add two double arguments, and return a double that is the sum of the values.

Enter your code below:
(With include statements)

```
int addDoubles(int numOne, int numTwo)
{
    return;
}
```

Expected	This Run	Correct?
	Program failed: error output is: program. c: In function 'addDoubles': program. c:3: warning: 'return' with no value, in function returning non-void	

Submit Reset

View other problems created by: [Roman, Justin](#)

Questions about this problem? Email [Roman, Justin](#)

To view more problems from this category, click [here](#).

Categories My Teachers Preferences Logout

Done

In order to submit a function, you must type it into the text area beneath the label “Enter your code below.” Be sure to include any includes your function will need to run. When you are happy with the function hit submit. Whether you’ve submitted answer is correct or not, the attempt is saved. If the answer you submit does not compile, then the results window will show the compiler errors and warnings (shown above). The function must compile with no errors or warnings before the site tests your results against the results stored from the teacher’s function.

Problem Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://babyhuey.cs.temple.edu:8080/CTester/problemProcess.jsp

Most Visited Getting Started Latest Headlines

Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options

Categories My Teachers Preferences Logout

Basic Arithmetic → Add Doubles

Description:

Write a function that will add two double arguments, and return a double that is the sum of the values.

Enter your code below:
(With include statements)

```
#include<stdio.h>

int addDoubles(int numOne, int numTwo)
{
    return numTwo;
}
```

Submit Reset

View other problems created by: [Roman, Justin](#)
 Questions about this problem? Email [Roman, Justin](#)
 To view more problems from this category, click [here](#).

Expected	This Run	Correct?
addDoubles(-5, 4) → -1	4	Red
addDoubles(0, 3) → 3	3	Green
addDoubles(-4, 4) → 0	4	Red
addDoubles(8e3, 8e2) → 8800	800	Red
addDoubles(-2, -2) → -4	-2	Red
addDoubles(12, 1) → 13	1	Red

Done

If the submitted function compiles, the answer your function returns for each test case will be compared with each test case's expected value. In the above example, the submitted problem was only correct for one test case. The correct test case is displayed in green. The incorrect test cases are shown in red. It may be useful to note, whether or not the solution your function provided was correct, the correct answers are shown. The correct answers are shown in the "Expected" column, in the following syntax: functionName(testValueArg1, testValueArg2, testValueArgN) → correctAnswer. Your answer is shown in the next box, in the "This Run" column. The "Correct" column is green for correct, and red for incorrect. If the answer is correct for all test points, the website shows that problem as completed in your teachers' list of problems for you.

Problem Page - Mozilla Firefox
http://babyhuey.cs.temple.edu:8080/CTester/problemProcess.jsp

Categories My Teachers Preferences Logout

Basic Arithmetic → C to F

Description:
Write a function that will convert temperatures in Celsius to Fahrenheit.
The formula is $F = 9/5 * C + 32$.

Enter your code below:
(With include statements)

```
int c_to_f(int temp_c)
{
    return (1.8 * temp_c + 32);
}
```

Expected	This Run	Correct?
c_to_f(0) → 32	32	✓
c_to_f(100) → 212	212	✓
c_to_f(-40) → -40	-40	✓
c_to_f(99) → 210	210	✓
c_to_f(1) → 33	33	✓

Submit Reset

View other problems created by: [Wolfgang, Paul](#)
Questions about this problem? Email [Wolfgang, Paul](#)
To view more problems from this category, click [here](#).

Done

In the example above, all the answers are correct, and all the answers are shown in green. Once this happens, your teacher can see the work you have done and when you completed it.

Project Future

What Did Not Get Done

- Have a button to show the current problem's solution
- Problem Navigation (i.e. previous, next, random)
- Pointer return types other than char
- Deactivate Account

What Could Get Done

- Better Navigation
- More Preferences (e.g. "Display My Email" checkbox)
- Allow teacher to decide if solution should be shown
- Allow teacher to comment on Student's completed problem
- Student's "my attempted problems" page
- Teacher assign problems to class, student view assigned problems
- Keep track of student's completed problems on categories.jsp and viewCategory.jsp with a check or color difference
- Forgot Password Link?
- Link to "How to Use our Website" (show example/walkthrough)
- *Allow teacher to view a student's attempted problems
- Edit/Remove categories
- *Put either an X or OK along with colors for the colorblind

Known Issues

Disregard for CSS File

Type of Error: Cosmetic

Browser Specific: Unknown

Frontend/Backend: Frontend

Frequency: Occasional

Description: The main page, categories.jsp, will occasionally disregard the linked CSS file causing unwanted bullets in the “class=blank” list items found under the categories.

Navigation Bars Not Centered

Type of Error: Cosmetic

Browser Specific: Internet Explorer 7 and earlier

Frontend/Backend: Frontend

Frequency: Always

Description: Internet Explorer will display the top and bottom navigation bars slightly askew to the right showing a bit of white on the left which causes web items to look unaligned. This issue is fixed in Internet Explorer 8. We implemented a temporary fix by making the whole background of the navigation bar the same color as that of the list items.

Password Textbox Size

Type of Error: Cosmetic

Browser Specific: Internet Explorer

Frontend/Backend: Frontend

Frequency: Always

Description: Internet Explorer will display the password textboxes a few pixels smaller than normal text textboxes of the same width.