Project 1A Specification

Project 1 Part A

Due Friday, 04/13/2018 by 11:59pm

Change history

Partners

The CS143 project may be completed in teams of one or two sutdents. The choice is up to each student. Partnership rules consist of the following:

- An identical amount of work is expected and the same grading scale is used regardless of the size of your team.
- If you work in a team of two, **choose your partner carefully**. Partners are permitted to "divorce" at any time during the course, and "single" student may choose to find a partner as the project progresses. However students from divorced teams may not form new teams or join other teams.
- Partners in a team will receive exactly the same grade for each project part submitted jointly.

If you have two students in your team, please make sure to submit your work jointly - do *not* turn your project in twice, once for each partner.

Scope

The primary purpose of this first "PHP warm-up" part is for us to provide you with a whole bunch of basic information, and to get everyone up-to-speed on our computing systems and the languages and tools we will be using. Those of you who have done some Web programming before, especially in PHP, may find this project part nearly trivial. Those of you who haven't will find it merely straightforward. With all that said, *please don't start at the last minute* -- as with all programming and systems work, everything takes a bit of time, and unforeseen snafus do crop up.

System Setup

We will be using VirtualBox to run the Linux operating system in a virtual machine. VirtualBox allows a single machine to share resources and run multiple operating systems simultaneously. Read our VirtualBox setup instruction and follow the instructions to install VirtualBox and our virtual-machine image on your own machine. In particular, please ensure that **a shared folder is set up correctly as `vm-shared`**. Otherwise, the Apache server in your virtual machine may not function properly.

The provided virtual machine image is based on Ubuntu 17.10, MySQL 5.7.21, Apache 2.4.27, and PHP 7.1.15. You will need to use the provided VirtualBox guest OS to develop and test all projects for this class. Your VirtualBox guest is essentially a Linux machine, which is a variant of Unix. Your guest is accessible from your host through secure shell (ssh) at 'localhost' port 1432 with username 'cs143' and password 'password'. Its Apache server is accessible from host at 'localhost' port 1438. If you are not familiar with Unix, now is the time to read the Unix Tutorial for Beginners to learn the basic Unix commands. **If you have any problems with the image, let the TAs know. We can provide you with an older image that work better.**

PHP Web Calculator

In this part of the project you will familiarize yourself with Apache2/PHP by building a small Web calculator application in PHP.

- **Step 1:** Review the W3CSchools PHP tutorial to learn basic PHP. Please read at least up to "PHP $_POST" page of the tutorial. You can test the examples in the tutorial by creating a php page in the `${HOME}/www/` directory in the guest OS, which is aliased (or symbolic linked) to the VirtualBox shared directory. All files in `${HOME}/www/` are served by the guest Apache server and are accessible at http://localhost:1438/~cs143/ from your host browser. Note that `${HOME}` is common Unix notation to refer to your home directory, which is `/home/cs143/` in our setup.
- **Step 2:** Play with our demo calculator here (access restricted to username 'project' and password 'demo') to understand what it does.
- **Step 3:** Implement your own calculator and make it available at http://localhost:1438/~cs143/calculator.php. At the minimum, your calculator application should satisfy the following requirements.
    1. It should support +, -, * and / operators and the evaluation of the input should follow the standard operator precedence (i.e., the operators should be left-associative and / and * operators have precedence over + and -).
    2. It should take both interger (like 1234) and real (like 123.45) numbers.
    3. It *does not* need to support parentheses. (As a side note, in case you took the compiler class before, you may remember that the correct handling of nested-parentheses requires more expressive power than regular expression, like context-free grammar.)

4. It should handle any errors gracefully. For example, even if the user input is invalid expression, the result page should not display raw PHP error message.
5. Your calculator should be implemented as a *single* .php page. Make sure that it does not include and/or read any other file (like CSS file, images or other HTML files).
6. For all links in your calculator, including form actions, you should use ***relative URLs*** instead of absolute. The reason is simple: depending on where we run your Web calculator, the hostname and the path of your calculator may change. So if you use absolute URL, you calculator may *break* when we try to grade it and we may have to give zero point for your work!
7. The calculator.php is more like a question-answering interface, and therefore you should use **HTTP GET** protocol to process the user input as suggested by W3C.

In implementing your calculator, you may find the PHP functions, `preg_match()` and `eval()` helpful. If you are not familiar with regular expression, read a tutorial on regular expression like Mastering regular expression in PHP. If you are not familiar with the HTML input forms, you may also find our tutorial on PHP input handling helpful.

Throughly test your code and make sure that it meets the above minimum requirements and runs correctly on our virtual machine. Note that this part of the project will be graded based on the functionality not on the look or style. As long as you meet the minimum requirements, you will get full credit for this project.

Late Submission Policy

Part 1A should be submitted the deadline to keep you on track, but it will not count for a grade, and late submissions will not receive any penalty.

Submission Instruction

Preparing Your Submission

Please create **a folder named as your UID**, put all your files into the folder, then compress this folder into a single zip file called "P1A.zip". That is, the zip file should have the following structure.

```
P1A.zip
 |
 +- Folder named with Your UID, like "904200000" (without quotes)
     |
     +- calculator.php
     |
     +- team.txt
     |
     +- readme.txt
```

Please note that the **file names are case sensitive**, so you should use the exact same cases for the file names. (For team work, use the submitter's UID to name the folder)

- `calculator.php`: Your `calculator.php` source code. Again, please make sure that all URLs in your code are ***relative***. You may get zero point otherwise.
- **`team.txt`**: A plain-text file (no word or PDF, please) that contains the UID(s) of every member of your team. If you work alone, just write your UID (e.g. 904200000).
  If you work with a partner, write both UIDs separated by a comma (e.g. 904200000, 904200001). **Do not include any other content in this file!**
- `readme.txt`: A plain-text file that includes any information you think is useful. If you have a partner, you may want to discuss briefly how the work was divided (For example, did you divide up the project and work separately? did you use pair programming? etc.).

Testing of Your Submission

Grading is a difficult and time-consuming process, and file naming and packaging convention is very important to test your submission without any error. In order to help you ensure the correct packaging of your submission, you can test your packaging by downloading this test script. In essence, this script unzips your submission to a temporary directory and tests whether or not you have included all your submission files. The script for Project 1A can be executed like:

```
cs143@cs143:~$ ./p1a_test <Your UID>
```

(Put your P1A.zip file in the same directory with this test script, you may need to use "chmod +x p1a_test" if there is a permission error).
You **MUST** test your submission using the script before your final submission to minimize the chance of an unexpected error during grading. Again, significant points may be deducted if the grader encounters an error during grading.When evertything runs properly, you will see an

output similar to the following from this script:

```
Check File Successfully. Please upload your P1A.zip file to CCLE.
```

Submitting Your Zip File

Visit the Project 1A submission page on CCLE and submit **only your "P1A.zip" file** electronically by the deadline. In order to accomodate the last minute snafu during submission, you will have 30-minute window after the deadline to finish your submission process. That is, as long as you start your submission before the deadline and complete within 30 minutes after the deadline, we won't deduct your grade period without any penalty.

FAQ

1. Q: Does the calculator need to handle negative/positive numbers?

   A: We will not use any positive sign before a number, but a negative sign needs to be supported. For example, we may test expressions like "3*-2" or "-2/-3" or "1+-1".

2. Q: how should we treat the fractional number: for example: would we treat ".123" as "0.123" or treat it as an invalid expression?

   A: All fractional numbers will have a leading zero in our test, but you are welcome to handle .123

3. Q: may TAs give us some testing cases to help us to get a 100% score?

   - -49 (Ans: -49)
   - 2+3+4 (Ans: 9)
   - 2*3*-4 (Ans: -24)
   - 2*-1*-2*-3 (Ans: -12)
   - 100-100/100 (Ans: 99)
   - 3/2+1/3 (Ans: close or equal to 1.83333333333)
   - 0/0 (NO excpetion shown on the page)
   - abcd (Invalid Expression)
   - one/two (Invalid Expression)

Last modified: Sunday, 8 April 2018, 2:24 PM PDT