Project 1 Part A

Due Wednesday, Apr. 11th, 20012 by 11:00pm

Change history

Partners

The CS143 project may be completed individually or in teams of two, 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 for individual and team projects.
- If you work in a team, **choose your partner carefully**. Teams are permitted to "divorce" at any time during the course, and individual students may choose to team up as the project progresses, however students from divorced teams may not form new teams or join other teams.
- Both partners in a team will receive exactly the same grade for each project part submitted jointly.

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 (Windows) (Mac) and follow the instructions to install VirtualBox and our virtual-machine image on your own machine. **(Particularly, if you are a mac user and see an "vboxnet0" error, please make sure you follow our mac instruction above step by step.)**

The provided virtual machine image is based on Ubuntu 7.10, MySQL 5.0.45, Apache 2.2.4, and PHP 5.2.3. 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. If you are not familiar with Unix, now is the time to read the Unix Tutorial for Beginners to learn the basic Unix commands.

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 `${HOME}/www/` are served by the guest Apache server and are accessible at `http://[guest IP]/~cs143/` from your host browser. Note that `${HOME}` is common Unix notation to refer to your home directory, which is `/home/cs143/` in our setup and `[guest ip]` is the IP of your guest machine, which is set up to be `192.168.56.20`.

- **Step 2:** Play with our demo calculator at http://oak.cs.ucla.edu/cs143/project/demo/calc/ to understand what it does.

- **Step 3:** Implement your own calculator and make it available at `http://[guest IP]/~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!

    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

To accommodate the emergencies that students may encounter, each team has 4-day grace period for late submission. The grace period can be used for any part of the project in the unit of one day. For example, a student may use 1-day grace period for Project 1A and 2-day grace period for Project 2B. Any single project part may not be more than 2 days late. Note that even if a team submits a project 12 hours late, they would need to use a full day grace period to avoid late penalty. If your project is submitted late, we will automatically use the available days in your grace period unless you specifically mention otherwise in the README file.

What to Submit

Please submit the following files electronically:

- Your `calculator.php` source code. Again, please make sure that all URLs in your code are ***relative***. You may get zero point otherwise.
- A README file that includes any information you think is useful. At a minimum, your README must contain your name, SID, and email address. If you are working as a team, make sure both team member's information is included, and discuss briefly how the work was divided (For example, did you divide up the project and work separately? did you use pair programming? etc.). **Your README file must be in PDF or PLAIN TEXT format. No Word documents, RTF, or any other format.**

Visit the Project 1A submission page to submit electronically by the deadline. In order to accomodate the last minite 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 11:29pm and complete the submission before midnight, 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)
- 2- -1 (Ans: 3 or say it's an invalid expression)
- 0/0 (NO excpetion shown on the page)
- abcd (Invalid Expression)
- one/two (Invalid Expression)