Programming Assignment 0
Due: January 15, 2018

0 Instructions

Submit on Canvas a tar file named lastname.tar (where lastname is your last name) that contains all of the required files. Don't submit any other files (e.g., test case or pyc files).

Here is an example of what to submit:

hampton.tar
  README
  my_class1.py
  my_class2.py
  my_class3.py
  problem.py
  another_problem.py
  ...

Note that Canvas might change the name of the file that you submit to something like lastname-N.tar. This is totally fine!

The grading for this assignment will be roughly as follows:

Task	**Points**
System Tools Warmup	participation credit
I/O Warmup	participation credit
Submit Warmup	participation credit
TOTAL	participation credit

1 Warmup Problems

Problem 1. System Tools Warmup

This problem will let you practice with some essential system utilities.

Before you start, you will need to set up a CIS Department systems account (if you don't already have one). You can stop by the department office in Deschutes Hall or use the following self-service link:

`https://systems.cs.uoregon.edu/wiki/index.php?n=Help.Account`

Reference the Lab 1 Notes for the terminal commands needed to complete this problem.

Log onto ix-dev and create a new directory named `CIS313`. Inside that directory create another directory named `pset0`. Go into the `CIS313/pset0` directory and copy the instructions for this problem, given by the following path, into your current directory:

`/home/users/hampton2/lab1_313/instructions`

Alternately, you can download the instructions from the following link:

`http://ix.cs.uoregon.edu/~hampton2/313/instructions`

View the contents of the instructions file to complete this problem.

Problem 2. I/O Warmup

Download the Programming Assignment 0 starter pack here:

`http://ix.cs.uoregon.edu/~hampton2/313/proj0_starter.tar`

Unpack it with the tar command (reference the Lab 1 Notes for help if needed).

The starter pack contains two Python 3 files: a Calculator class and a driver program. You won't be required to use Python to complete the course assignments, but it's strongly encouraged! Python is a great language to know right now for succeeding with tech interviews.

The calculator class has already been written for you. It implements the following (public) methods:

`add(X)`: Takes a single integer argument and adds it to the running total. Should be amortized $O(1)$ time.

`mult(X)`: Takes a single integer argument and multiplies the running total by X. Should be amortized $O(1)$ time.

`seen(X)`: Takes a single integer argument and returns a boolean indicating whether the running total has ever been equal to X. Should be amortized $O(1)$ time.

The driver program has also already been written for you! It takes a single command-line argument, which will be a filename. The input file will contain instructions for calculator operations. The first line of the input file will be an integer $1 \leq N \leq 10^3$ giving the number of instructions. Following will be N lines, each containing an instruction. The possible instructions are:

`plus X`, where $-10^6 \leq X \leq 10^6$ is an integer: For this instruction, add X to the running total. Output the running total.

`times X`, where $0 \leq X \leq 10^3$ is an integer: For this instruction, multiply the running total by X. Output the running total.

`seen X`, where $-10^7 \leq X \leq 10^7$ is an integer: For this instruction, give an indication of whether the running total has ever been equal to X. Output `YES` if the running total has ever been equal to the given value, else output `NO`.

All output should be to STDOUT. Each piece of output should be separated by a newline.

Example input file:

```
9
times 200
plus 1
plus 2
seen 5
seen 0
times 5
seen 5
seen 3
seen 15
```

Example output:

```
0
1
3
NO
YES
15
NO
YES
YES
```

The starter pack also includes sample input and output files. Your programming assignments will be graded on ix-dev, so you want to be sure that your code works on there. Send your source files over to ix-dev (e.g., using the `scp` command) and run them there. The Python 3 interpreter on ix-dev is invoked as `python3`.

There is a system utility called `diff` that is super helpful for seeing if two files match. The `diff` utility will even catch whitespace differences that are easy to miss.

It's often convenient to save the output from a program to a file. We can do that with the *redirection* operator:

```
python3 driver.py sample.input > my_output
```

Note that file extensions (such as `.txt`) aren't as strict on a UNIX-like system as they are on, say, Windows. We can typically use whatever extensions we want. If you want the system to tell you what kind of file you've got, use the `file` utility:

```
file my_output
```

Now, let's compare `my_output` and `sample.output`:

```
diff my_output sample.output
```

If you run the `diff` command and don't see anything in the terminal, that means it was a perfect match! You'll notice, however, that we don't have a perfect match. Use a reference to help you learn how to read the results of `diff`.

The Calculator class and driver class were implemented for you, but there are two bugs that need to be fixed! The bugs were caught during code review, so read the files and implement the two fixes that are suggested in the comments.

Once you've got a solution that produces a perfect match according to `diff`, you're done with this problem.

[Just for fun: if you just want to compare the output of a program with a file, it's not necessary to create a temporary file. The *pipe* operator allows you to use the output from one command as the input to another command:

```
python3 driver.py sample.input | diff - sample.output
```

This is just a brief intro to UNIX system utilities. Learning to use these tools to automate system jobs will save you a lot of time down the road!]

Problem 3. Submit Warmup

In this problem, you'll get practice bundling your submission into a `tar` file and uploading it to Canvas.

Navigate to your `CIS313/pset0` directory on ix-dev. Create a new directory called `submit`.

Copy into this new directory (either from elsewhere on ix-dev or from your personal computer) everything that you need to turn in. For this assignment, it's the following files:

```
problem1_solution
calculator.py
driver.py
```

It's a good idea to go through the steps from Problem 2 to test your solution again. It's easy to avoid accidentally submitting the wrong files.

Once you're sure you've got the correct files, write a README for your project. Remember, you can use the following command to create a new file called README and open it in the nano editor (feel free to use a different editor if you prefer):

```
nano README
```

Your README should contain your name and exact commands that can be invoked on ix-dev to run your solution for each problem. If your solution takes an input file, just use a dummy name. For example, your README for this project should look something like this:

```
Andrew Hampton

Problem 2: python3 driver.py some.input
```

If you use a language that requires compilation, your README should also include the exact commands to build your project on ix-dev.

Finally, create a `tar` file that contains everything you want to submit. In the filename `lastname.tar`, replace `lastname` with your last name:

```
tar cvf lastname.tar problem1_solution calculator.py driver.py README
```

You can check that the file you've created is actually a tar file by using the `file` command:

```
file lastname.tar
```

Now, just copy the tar file back to your personal computer and upload it to Canvas for Programming Assignment 0.

This is how you will turn in all of the programming projects for this course (and most of your upper division CIS courses). It's important to follow these instructions because it allows parts of the grading process to be automated. You will lose a substantial amount of credit for an assignment if you don't follow the submission guidelines or submit the wrong files.
