CS 256 – Programming Languages & Translators
Homework Assignment #8

- This assignment is **due by 8 a.m. on Monday, December 11th.**
- This assignment will be worth **4%** of your course grade.
- You are to work on this assignment **by yourself**.

Basic Instructions

For this assignment you are to write an **"automated grader script" in bash**. Basically, your script is to take a zip file containing student submissions of **Lisp** programs and execute each program for each input case (which is specified in a given set of input files), comparing the student's output to the respective (given) expected output, and finally creating an output file that has each student's score for the "entire assignment."

Consider the following example that describes more specifically what your script should do.

You will be given a file named **submissions.zip** that contains a Lisp file for each student submission. Each file in the zip file is named with the student's last name and first initial. For this example, assume submissions.zip just contains 3 files: simpsonh.lisp, and duckd.lisp, and leopoldj.lisp.

You will be given a file named **sampleInput.zip** that contains a text file for each input case that we want to test a student's program on. For this example, assume sampleInput.zip just contains 2 files: input1.txt and input2.txt. These files will contain Lisp code that we want to call to test a student's program. For example, suppose the students were supposed to write a function named *f* that takes two (presumably integer) parameters, adds them, and returns the result. If we were testing these programs, input1.txt might contain ***(print (f 2 2))*** and input2.txt might contain ***(print (f 100 -3))***.

You also will be given a file named **expectedOutput.zip** that contains a text file for each input file that is in sampleInput.zip; these files are named the same as the sample input files, except they have **.out** on the end of the name. For this example, our expectedOutput.zip would contain 2 files: input1.txt.out (which would contain **4**, the result of adding 2 and 2) and input2.txt.out (which would contain **97**, the result of adding 100 and -3).

The first thing that your script should do is unzip the sampleInput.zip file into a directory (we'll call it sampleInput), unzip the expectedOutput.zip file into a directory (we'll call it expectedOutput), and unzip submissions.zip into a directory (we'll call it submissions).
For documentation on the linux/unix *unzip* command, see:
https://linux.die.net/man/1/unzip

Next you'll need to 'process' every entry in the submissions directory. This requires that, for every input file in your sampleInput directory, you invoke **gcl** on that submission with that input case, compare the output to the respective output file in the expectedOutput directory, and keep a total of how many input cases the student got right out of the total number of cases tested. For each student you process, you should output this statistic (as a percentage), as well as the student's last name and first initial to a text file named **grades.txt**. For example, if simpsonh got 1 of the 2 input cases correct (50%), duckd got 0 of the 2 input cases correct (0%), and leopoldj got 2 of the 2 input cases correct (100%), then your grades.txt file should look like the following:

simpsonh, 50
duckd, 0
leopoldj, 100

In order to compare the student's output to the expected output, you may want to write a student's output to a file and then use ***diff*** on the 'actual' and 'expected' files; see http://ss64.com/bash/diff.html

You may assume that no student's submission will contain code that will "crash" gcl, and hence "crash" or "hang" your script ☹

Your script will need to invoke gcl, loading the student's Lisp file and telling gcl what to evaluate (the latter of which is specified ***partially*** in the sample input files). The command for doing this is: **gcl –load *filename* –eval "*lispExpression*" .** If you wanted to redirect the output of that command to a file, you would add **> *outputFilename***. In order to have the Lisp interpreter exit after evaluating lispExpression, you must add **(quit)**. To add that to *lispExpression*, you must wrap it in a **(progn …)** expression. So, as an example, if you wanted to run the submission leopoldj.lisp, and the input1.txt file contained *(print (f 2 2)),* and you wanted the output to go in a file named leopoldj_input1.txt.out, you would need to do:

gcl –load leopoldj.lisp –eval "(progn (print (f 2 2)) (quit))" > leopoldj_input1.txt.out

Sample files are provided on Canvas. **However, they are <u>NOT</u> necessarily the files we will use for grading your homework!** To make things somewhat simpler for you, you may assume that all the Lisp function inputs and expected outputs will be integers; you don't have to worry about real numbers, strings, etc.

But wait…there's one more thing you need to do! Believe it or not, sometimes students hard-code into their programs outputs for specific inputs that they expect the instructor will test their program on. We want your script to try to detect this by doing the following. If the student's submission produces correct output for an input case, your script must look at his/her source code to see if you can find that output (in any context – it doesn't have to be in a *print* statement); ***grep*** would be a good tool for doing this. If this occurs, we want you to output an asterisk ***** next to the student's score in the

grades.txt file. This will inform us that we should manually grade that student's program! ☹

Warning: Your bash script **must** execute on one of the campus Linux machines. Various operating systems (e.g., Windows, Mac OS, etc.) do things differently, particularly when it comes to file permissions, directory management, and versions of *sh*. So you should **test your script on the campus Linux machines before you submit it for grading!!!**

What to Submit for Grading:

Via Canvas you should submit only your *sh* file. Name your *sh* file using **your last name followed by your first initial** (e.g., Homer Simpson would name his file **simpsonh.sh**). You can submit multiple times before the deadline; only your last submission will be graded.

WARNING: If you fail to follow all of the instructions for submitting this assignment, your homework will NOT be graded!!!

The grading rubric is given below so that you can see how many points each part of this assignment is worth.

Functionality	Points Possible	Mostly or completely incorrect (0% of points possible)	Needs improvement (50% of points possible)	Mostly or completely correct (100% of points possible)
Script unzips submissions.zip into a separate directory	2			
Script unzips sampleInput.zip into a separate directory	2			
Script unzips expectedOutput.zip into a separate directory	2			
Script has a loop to process each submission	8			
Script has a loop to process each input case for each submission	8			
Script correctly calls gcl for each submission for each input case	8			
Script correctly compares each student's output to expected output	3			
Script correctly keeps score for each student	4			
Script correctly computes each student's percentage for 'assignment'	3			

Script correctly creates grades.txt containing 'assignment' percentages	5			
Script correctly detects 'suspicious' programs and marks them in grades.txt	5			