CS 3500 – Programming Languages & Translators
Homework Assignment #8

- This assignment is **due by 11:59 p.m. on Wednesday, May 8th.**
- This assignment will be worth **3%** 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 CS 1570 student submissions of **C++** programs that are supposed to compute the number of combinations of *n* items taken *r* at a time (i.e., *C(n, r)*), using *n* and *r* as command line arguments (e.g., you would expect to run an executable as ***a.out 10 5*** and have it output **252**)[1].

Your bash script must do all of the following:

(1) **Create a directory named submissions** in the current working directory.

(2) **Unzip** a file named **submissions.zip** (a sample of which is posted on Canvas) into the directory named submissions. This file contains the students' programs. For documentation on the linux/unix *unzip* command, see: https://linux.die.net/man/1/unzip

(3) Each student submission file will have a name of the form: 1 or more letters, an underscore, 1 or more digits, an underscore, 1 or more digits, an underscore, the student's last name and first initial, a period, and *cpp* For example, there could be a file named *leopoldjennifer_123_456_leopoldj.cpp* You are to **rename each file** to just the student's last name and first initial, a period, and *cpp* (e.g., you would change the previously given file name to just *leopoldj.cpp*). Additionally, there could be a dash followed by a number before the *.cpp* if the student submitted multiple times (e.g., *leopoldj-3.cpp*). You must remove that as well (e.g., we just want *leopoldj.cpp*).

(4) For each submission, **compile it using g++.**

(5) For each executable, **run it** on command line input of **10 5** and see if the output is **252**. You can assume that the students' programs are expecting command line input.

(6) **Output** each student's last name and first initial to a file named **grades.txt** (which you create in the current working directory). If the student's output was correct, output a comma after their name and "correct"; otherwise, output a comma after their name and "incorrect". If the student's program contains 252, after "correct", output a space and "HARD-CODED OUTPUT!".

[1] See https://www.calculatorsoup.com/calculators/discretemathematics/combinations.php

You may assume that no student's submission will contain code that will "crash" g++ or not compile, and hence will "hang" your script ☹

Note that the sample files provided on Canvas were created on Windows and may contain characters like '\r' that Unix/Linux doesn't like; **run dos2unix** on them before you use them. **Also note that these are <u>NOT</u> necessarily the files we will use for grading your homework! Your script must work for <u>any number</u> of submissions; do not assume you will get exactly the same number as are posted in the sample file we put on Canvas!**

<u>Warning</u>: Your bash script **<u>MUST</u>** execute on one of the campus Linux machines. Various operating systems (e.g., Windows, Mac OS, etc.) do things a bit differently, particularly when it comes to file permissions, directory management, and versions of *sh*. Therefore, you should **test your script on the campus Linux machines before you submit it for grading!!! Excuses such as "but it runs fine on my Mac" will not be accepted** ☹

<u>What to Submit for Grading</u>

Via Canvas you should submit <u>only</u> your *sh* file. Name your *sh* file using **your last name followed by your first initial** (e.g., Bojack Horseman would name his file **horsemanb.sh**). You can submit multiple times before the deadline; only your last submission will be graded.

<u>WARNING</u>: If you fail to follow all of the instructions for submitting this assignment, your homework will <u>NOT</u> 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 correctly unzips submissions.zip	2			
Script strips off unwanted chars from each submission file name	10			
Script has a loop to process each submission	8			
Script correctly compares output of each student's program to expected output	6			
Script correctly detects hard-coded answer in student programs	6			

Script correctly creates grades.txt with each student's name	6			
Script correctly creates grades.txt with "correct" or "incorrect" for each student	6			
Script correctly creates grades.txt with "HARD-CODED" message as appropriate	6			

50