BASH GRADER

(TA: Saksham Rathi, CS 108)

This project is intended to create a **csv file manager** and **interpreter**. A csv (comma separated values) file is a text file which allows data to be stored in table structured format. So, for every row, the individual values will be separated by commas.

Suppose you are provided with a large number of such files. Each such file has data for a student for a particular exam. The format of the data is as follows: <Roll_Number,Name,Marks>. Every csv file will have a heading and then rows of data for various students. For running your bash script, you need to type: "bash submission.sh <command> <any other extra arguments(if needed)>". Here are some of the sample files for this project.

You need to accomplish the following tasks using submission.sh. Below, we have specified the commands and what needs to be accomplished via them.

1. **combine:** You need to create a main.csv file which contains the data for all the exams so far. So, on running the command , your bash script should iterate over all the csv files in that directory and should combine to create main.csv. So, let's suppose you have three files: quiz1.csv, quiz2.csv, endsem.csv. Then your main.csv should have the following columns: <Roll_Number,Name,quiz1,quiz2,endsem> (Assume the roll numbers to be case-insensitive. Therefore, "B" is the same as "b"). So, we add a column corresponding to every file in the main.csv file. Also, note that the sequence of students in every file might be different. So, main.csv can have the rows in any sequence. Moreover, some student might be absent in a particular exam, so his/her name might not be present in that particular csv file, for such cases your main.csv should have "a" instead of the marks. For example, if Saksham was absent in quiz1 and secured 6 and 26 in quiz2 and endsem respectively. So, in main.csv, his entry should be: "22B1003,Saksham,a,6,26".

The number of exams held so far are also not fixed. You need to update main.csv for every other csv file present in that folder. Overall, the students in main.csv will be the union of all the students in every other csv file. You can assume that no two students have the same roll number. (But they may have the same names!)

2. **upload:** There should be a provision of uploading/importing new csv files. So, on running the command "bash submission.sh upload ~/Desktop/project.csv, your bash script should copy the corresponding file into the script's directory. Now, whenever you run the "combine" command next, your main.csv file will be updated accordingly.
3. **total:** On running the command "total", your script should create a new column (with heading "total") in main.csv, which stores the total of every student. (with absent being treated as 0). Remember, your script should still remain capable of running combine for adding new files in main.csv. The total in such cases should be changed accordingly. (So, whenever you run the combine again, the total should be updated accordingly.)
4. You have already learnt about Git and how it manages versions of a file through commits. We will incorporate a smaller version of Git here in our script. So, basically we will be storing all the versions of our current folder in some other folder. You need to implement three commands for git:
    - **git_init:** This command initializes the remote directory. So, suppose you type, "bash submission.sh git_init ~/Desktop/bc", then your script will mark the folder "bc" on Desktop as the remote repository. (You need to create such a folder, if it does not exist already!)
    - **git_commit:** This command copies the current version of all the files to that remote directory. If git_init has not been executed already, then this should print an error message and stop executing the script. (You need to see how you will be storing the remote repository path!). This command can be executed as follows: "bash submission.sh git_commit -m "commit-message". Every commit should generate a random number (of 16 digits) (also known as hash value). This number will act as an id of the commit. All these hash values and the corresponding commit messages should be stored in a .git_log file present in the remote directory. Also, you need to print the names of all the files which were modified after we did the last commit.

- **git_checkout:** This command reverts our current directory back to the commit message we specify. This command can be executed as "bash submission.sh git_checkout -m "commit-message" or "bash submission.sh git_checkout <hash_value>". Also, the user may not give the complete hash value, even if some prefix of that value is provided, the script should checkout to that commit. (You can return a conflict if two different hash values start with the same prefix.) (An easy way to do this is by storing every commit and files at that commit in the git_init repository. And, then when we do git_checkout, just copy all those files into the current directory!)

5. **update:** What if the corresponding TA wishes to increase marks after cribs? Most of you will agree that there must be such provision. So, on running the command "bash submission.sh update", the script should take the student's name, roll number and various marks as input (not as command line arguments!) (roll number should be the primary match criteria, your script should show some error message in case name does not match the roll number row) and update the main.csv file accordingly. (This update function should also change the individual files, so that next time you do combine, the marks are shown correctly.) (The way you take inputs is totally upto you. Like you can ask the user whether he wants to update this exam's marks or not. Or the input can also be taken in a specific format. For example: <exam_name marks>)

Also add feature to add number to all student for bonus question

Customizations:

Everyone should respect creativity. So, just to make your lives more horrible, you should also add some customizations to your bash script. Here are some ideas:

1. **Include more statistics:** Like, you can include commands to display various other statistics such as mean, median, standard deviation and so on. You can also add commands to show the marks of a particular student.
2. **Graphs:** For better visualization, you can display graphs related to the performance of students on various parameters. You can use python and modules for this.

3. **More git commands:** There are only 3 compulsory git commands, you can add more to make your script even more powerful. You can also make the current commands more efficient. For example, when you commit, there is no need to store every version of your files, you can just store the changes you did at every step. This can be done through **diff** and **patch** commands. Check this for reference: https://www.scaler.com/topics/linux-patch/#

These are just some of the possible ideas. You are encouraged to think of more innovative customization options.

Marks Distribution (15 Marks):

- All the compulsory commands: 8 Marks
- Customization: 5 Marks (This also carries a significant weightage towards your total, so you are expected to include as much extra stuff as you can think of.) (Also, customizations will be relatively graded.)
- Modularised Code (Modularisation means you should divide your code either into functions or into multiple files): 1 Mark (Also your code should be heavily commented!)
- LaTeX Report: 1 Mark

Vivas will be taken to test the overall correctness of your code. They don't have any separate weightage.

LaTeX Report:

The report must contain the following points:

- Basic Logic of the script: How your script functions on seeing a variety of commands.
- Utilities you used:What all programming languages you used in this project. (Bash, sed awk for basic commands. For the customizations, you can even use python, c++, HTML and so on.)

- Customizations: This section will help us differentiate your project from others. You are free to do as many customizations as you can do.
- Video demonstration (Optional: no marks for doing this): You can make a short video demonstrating how your script runs various commands. Then this video can be attached as a link with this project report.
- References: What all websites/books/resources you had used to create your project.

A sample report has been attached as an exercise with LaTeX tutorial slides. Your report can have even more stuff. (You can also add images, if you wish to demonstrate something.)

Upload

You need to submit all your bash script files including the LaTeX report. There are two main files. Your bash script will be present in "submission.sh" and LaTeX report will be present in "report.pdf". You can add some other files too. (Maybe for modularisation, sed, awk or customizations.)

Submission Instructions:

1. Create a folder named "submission". (Lowercase)
2. Place all your files (.sh, .sed, .awk, .py, .pdf or whatever files you wish to submit) inside that folder.
3. Zip this folder. The zip file should be named as "submission.zip".
4. Submit the "submission.zip" and report.pdf on vLab under the "Bash Grader" activity of "PROJECT" lab. For copying these two files, use vLab "Open Directory feature". The deadline for doing this is 26 April 11:59 pm. Also, after submitting, download your submissions to verify.

Although you can have as many files as you wish to, submission.sh (inside submission.zip) and report.pdf must be present. Not following the submission instructions or naming conventions will result in some penalty being imposed.

All the basic commands have to be completed using Bash only. (Although you are free to use sed and awk.) For customizations, you can use python scripts. Also, your script (submission.sh) will be present in the same directory as that of the csv files.

The use of AI tools is not prohibited. But, you should be able to explain any part of your code during your viva. Also, your code will be checked for plagiarism. And, if it is found that your code matches significantly with some other code, then you will be heavily penalized. (The software we will use is smart enough to ignore changed comments or different variable names.) So, basically any kind of discussions are allowed, but please don't exchange codes (and then later risk your lives.)

For any queries related to this project, please contact ***Saksham Rathi***, either through WhatsApp on 6375663539 or drop an email at 22b1003@iitb.ac.in
Also, any doubts related to how you should implement your projects are also welcome!

ALL THE BEST!!!