Indian Institute of Technology
Bombay

Department of Computer Science

Bash Grader Report

Tanishq Mandhane
23B0986

Supervised by
Prof. Kameswari Chebrolu & TA Saksham Rathi

April 28, 2024

Contents

1 Objective **2**

2 Introduction **2**

2.1 Basic Commands . . . 2

2.1.1 combine . . . 2

2.1.2 upload . . . 2

2.1.3 total . . . 2

2.1.4 update . . . 3

2.1.5 git_init . . . 3

2.1.6 git_commit . . . 3

2.1.7 git_checkout . . . 3

2.2 Customization . . . 4

2.2.1 Modified combine . . . 4

2.2.2 upload . . . 4

2.2.3 git_commit . . . 4

2.2.4 git_log . . . 4

2.2.5 git_status . . . 4

2.2.6 git_newbranch . . . 4

2.2.7 git_branch . . . 5

2.2.8 git_checkout branch . . . 5

2.2.9 examstats . . . 5

2.2.10 stats . . . 6

2.2.11 grades . . . 6

2.2.12 calculate_cpi . . . 7

2.2.13 Error Handling . . . 9

3 Logic of commands **9**

3.1 combine . . . 9

3.2 upload . . . 9

3.3 total . . . 9

3.4 update . . . 10

3.5 git_init . . . 10

3.6 git_commit . . . 10

3.7 git_checkout . . . 11

4 Utilities **11**

1 Objective

This project is intended to create a csv file manager and interpreter using ***Bash Script***.

2 Introduction

All the commands will run using submission.sh (Usage : `bash submission.sh <command> <any other extra arguments(if needed)>`). For each command Error Handling have been written such that all the commands and arguments given are correct, And if they are correct then submission.sh will run respective files present in same directory otherwise will give respective Error message on terminal.

2.1 Basic Commands

2.1.1 combine

- **Usage:** `bash submission.sh combine`
- This will combine all the non empty csv files (except main.csv) into main.csv which will include marks of students in all exams separated by comma.
- Students who are absent in a particular exam will get 'a' in the corresponding student-exam cell in main.csv.

2.1.2 upload

- **Usage:** `bash submission.sh upload <File_path>`
- File present at that path will be copied to the working directory and then we can run combine again to update our main.csv accordingly.

2.1.3 total

- **Usage:** `bash submission.sh total`
- This will add a new column 'total' in main.csv which will perform total of all marks of a student and append it in the student's row.
- For students who were absent in a exam, their marks in that exam will be taken 0 for totaling.

2.1.4 update

- **Usage:** `bash submission.sh update`
- This will ask for roll number and student's name and if given name doesn't match with the student's name then error will be shown up.
- In exam in which he/she is present it will ask whether to update marks or not .And in exams in which he/she is absent ,will ask whether or not his entry is to be added .

2.1.5 git_init

- **Usage:** `bash submission.sh git_init <Path of remote directory>`
- It will make remote directory at the given path specified in second argument.
- Inside remote directory each folder will signify each branch with each containg its own .git_log file which will contain commits hash value,message and date and time.

2.1.6 git_commit

- **Usage:** `bash submission.sh git_commit -m "<commit message>"`
- This part will store all the current version of csv files into folder whose name will be randomly generated 16 digit number (Hash value). This folder will be stored in current branch name folder.
- All these hash values and the corresponding commit messages will be stored in a .git_log file.It will also print modified files, new files and deleted files.

2.1.7 git_checkout

- **Usage:**
  1. `bash submission.sh git_checkout -m \commit-message"`
  2. `bash submission.sh git_checkout <hash_value>`
- This command reverts our current directory back to the commit having corresponding commit message or hash value.
- Some prefix of the hash value will also suffice until there are no more than one commit having same prefix. In case this happens then it will give error message.

2.2 Customization

2.2.1 Modified combine

- **Usage:** `bash submission.sh combine <files name>`
- The basic combine just takes one argument combine and combines all the csv files into main.csv .
- But in modified combine along with "combine" we can also give name of those exams which we only want to combine as arguments.

2.2.2 upload

- user can also upload folder instead of only files

2.2.3 git_commit

- Along with Modified files, It also prints out Deleted files and New files.

2.2.4 git_log

- **Usage:** `bash submission.sh git_log`
- It prints out all the details of commits which are made till now in the current branch.These details are stored in .git_log file.
- Detail includes hash value, commit message, and date and time.

2.2.5 git_status

- **Usage:** `bash submission.sh git_status`
- It prints out name of csv files under three categories -Modified files ,New files and deleted files. These are based on comparing files at that instant with files present in head commit.

2.2.6 git_newbranch

- **Usage:** `bash submission.sh git_newbranch`
- This command is use to create a new branch from.
- If the branch name already exist then it prompt that The given name already exists.
- If till now no head commit (means no commit have been made till now) is present then it changes the name of current branch ,doesn't create the new branch.

2.2.7 git_branch

- **Usage:** `bash submission.sh git_branch`
- It prints out all the names of branches which have been made.
- There will be a "*" mark in front of branch name on which we are currently working.

2.2.8 git_checkout branch

- **Usage:** `bash submission.sh git_checkout branch <Branch_Name>`
- It is use to checkout from one branch to another branch. It will checkout to last commit made in that branch.

2.2.9 examstats

- **Usage:** `bash submission.sh examstats`
- It displays options for selecting an exam based on CSV files present in the current folder, and then allows the user to choose an exam for detailed analysis,

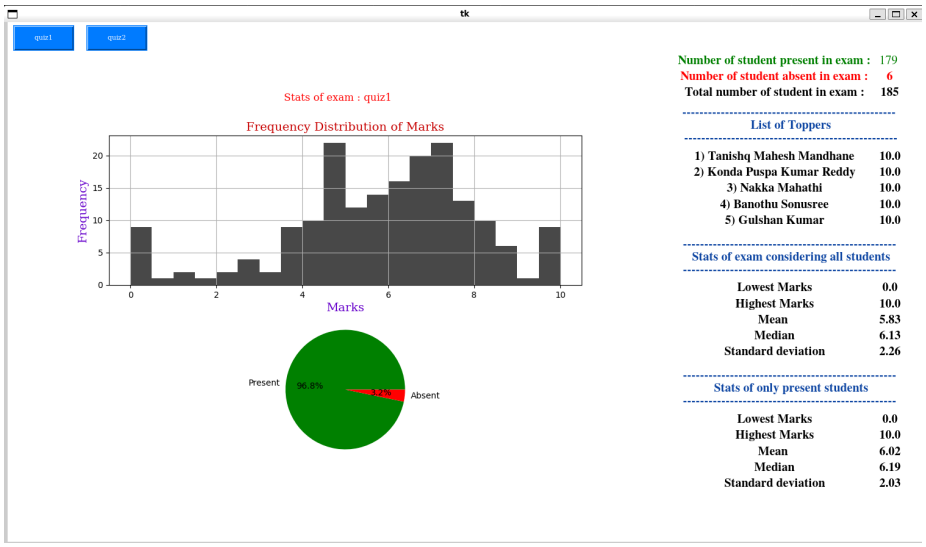

Figure 1: examstats command

2.2.10 stats

- 1. **Usage:** `bash submission.sh sats student <Roll_number>`
  2. **Usage:** `bash submission.sh stats compare <Roll_number_1> <Roll_number_2>`

- first command is use to see stats of that particular student whose roll number is given in 3^{rd} argument in all the exams.
  e.g On running `bash submission.sh stats student 23b0986`

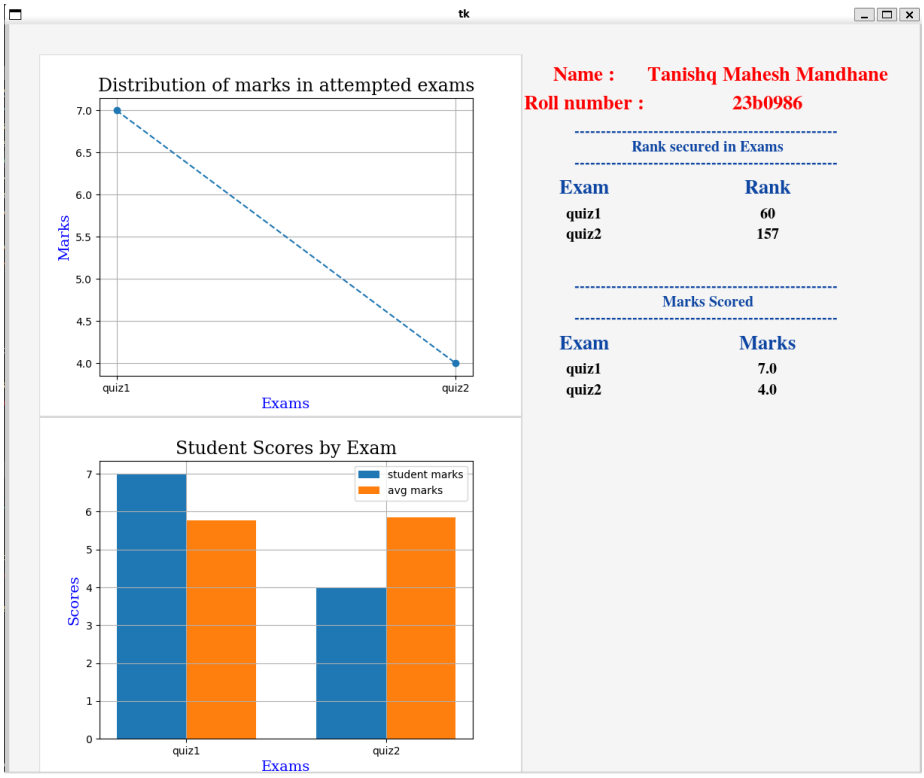

Figure 2: stats student command

- Second command is use to compare stats of two students whose roll numbersare given as 3^{rd} and 4^{th} argument.
  e.g On running `bash submission.sh stats compare 23b0986 23b0989`

2.2.11 grades

- **Usage:** `bash submission.sh grades`

- It displays options for selecting an exam based on CSV files present in folder and then allow user to choose an exam for which user wants to see

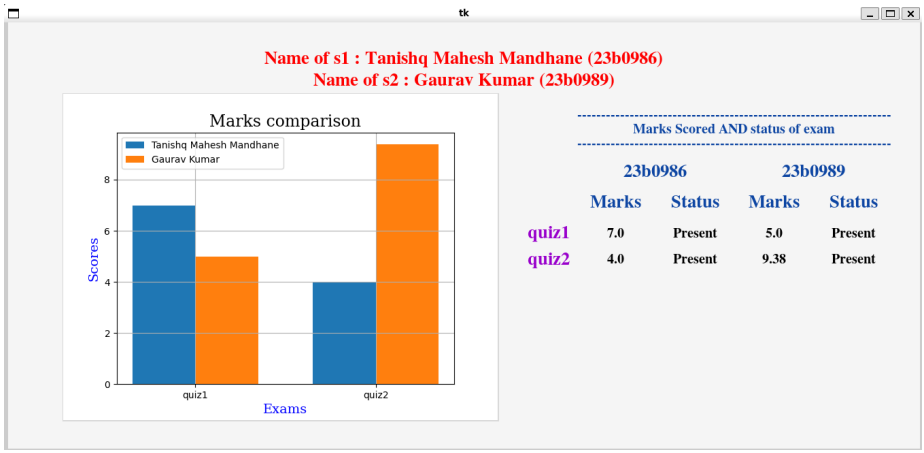

Figure 3: stats compare command

frequency-grade distribution.

- When you click any exam name then it also generate **grades.txt** file containg grades received by students in table format.
- This grading is done on the basis of algorithm written in **grader.py**

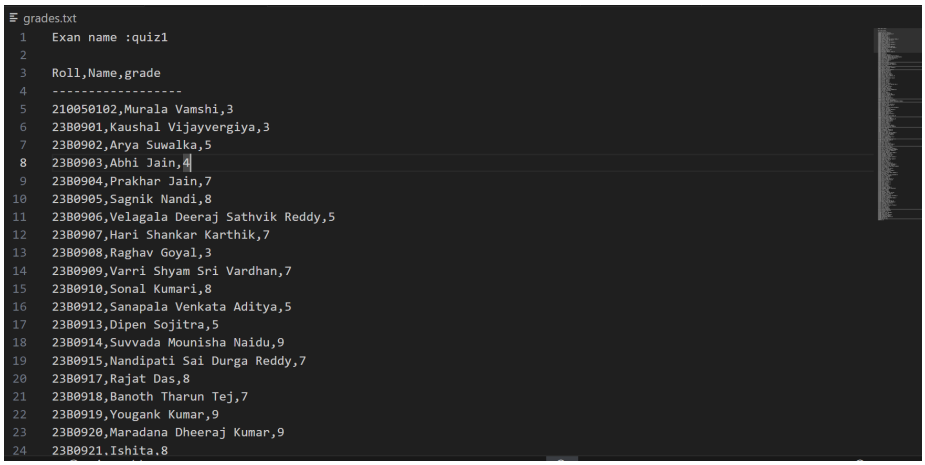

Figure 4: grades.txt

2.2.12 calculate_cpi

- **Usage:** `bash submission.sh calculate_cpi`

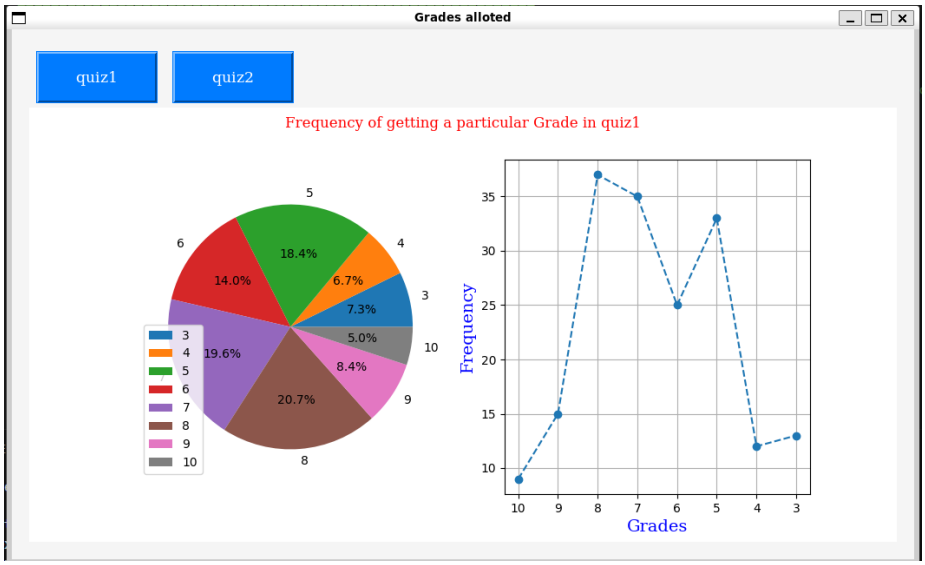

Figure 5: grades stats

- Its an GUI which I have created just for fun stuff use to calculate CPI of student. For each course it is use to take name of course, grades received and credit of that particular course.
- We can also add more course using Add Course button.And After filling all entries click calculate cpi button.

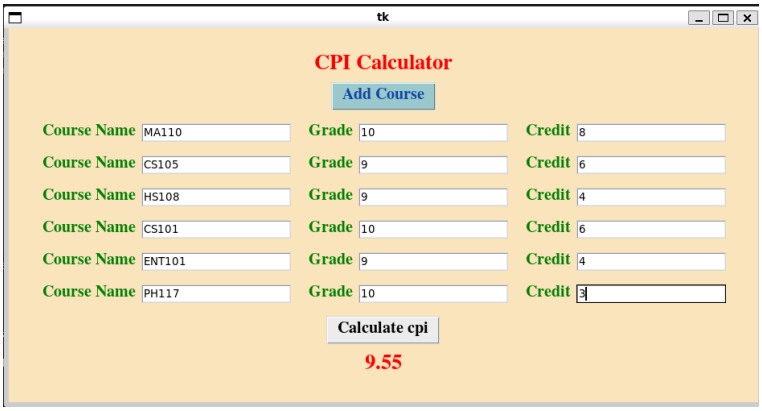

Figure 6: calculate cpi command

2.2.13 Error Handling

- Extensive Error handling have been implemented .

3 Logic of commands

3.1 combine

In this, submission.sh will call combine.sh in which our main logic is written. 'results' is an array which will store name of exams which we need to combine. If the number of arguments is 1 then 'results' will contain all exam name corresponding to all file and if its more than one then 'results' will contain exam name which are written in arguments.
First it iterates through all files in array 'results' and stores roll numbers in array name 'Roll_Numbers', and simultaneously stores name of that student in associative array 'Name' with roll number as key and name as value.(Since roll numbers are case insensitive I have used lowercase of roll numbers as key).
Firstly It creates heading for main.csv, by appending exam names in string initially containing "Roll_Numbers,Name". Then echo it into main.csv .
Then It iterates through all the roll numbers stored. For each roll number I initialize a string name 'performance' containing "{Roll_number},{Name}" of that student .And for each roll number it iterates through all the exam files .If the roll number is present ,then it will append ",{marks}" in performance otherwise will append ",a".We will get marks by using cut command with delimeter "," and field 3. We will check that student is present or not using grep command.If grep result is non empty then it is present and if it isempty then that roll number is not present.

3.2 upload

It just copy the file/directory at path given in 2^{nd} argument to current directory.

3.3 total

If the first argument is total then it will run awk file total.awk on main.csv and will store content in temporary file .main and then copy content of .main to main.csv and then deletes .main .

logic of total.awk :-If main.csv contain total as its last column then it will add marks from 3^{rd} field to $(NF-1)^{th}$. And if total column is not present then it will create one new column and for sum it will take sum from 3^{rd} field to NF^{th} field.

3.4 update

If the first argument is update then submission.sh will call update.sh .
It will first ask for name and roll number of student whose marks have to be updated. If given name doesn't match the name given in files of corresponding roll number then it will prompt error.
It iterates through all the csv files except main.csv. For each file it checks if student with given roll number is present or not(using grep). If the student is present then it asks whether user want to update marks in given exam in format of (y or n). If user wants to then it ask for updated marks and change marks using sed in that exam.csv file only. If the student is absent then it promt that the given student was absent ask whether to add its entry in given file in same format of (y or n). If user wants to then it ask for marks and add its entry in format "rollnumber,name,marks" into exam.csv file.
Finally it uses modified version of combine to combine those exams which were initially present in main.csv.

3.5 git_init

It first calls the git_init.sh script with address of remote directory as first argument. Then it makes folder according to given path.Next it makes sub folder name "master" which will be our initial branch in git. Each branch will have its own .git_log file where we will store commit message and hash value and date and time.
Simultaneously we will make three txt file which will be use to store value which we will use many times.These are:-

1. .currentbranch.txt-This file will be created in current working folder which will store current branch at each moment. Initially it will store "master".

2. .gitrepoloc -This file will be created in current working folder which will store path of remote directory.

3. .head.txt -This file will be created in remote directory which will store hash value of head commit. HEAD is a symbolic reference pointing to wherever you are in your commit history.

3.6 git_commit

If the syntax of ussing git_commit command is correct then it first calls the git_commit.sh script. It first store the path of remote directory in variable "path",value of current branch in "currentbranch" and generates a 16 digit randomly generated number and stores it in "Hash_value".
Then it initializes 3 arrays for storing modified files,deleted files and new files. If file present in current folder is also present in head commit then it checks for diff of both files version and if it is not present then it is New file. If diff is non empty then it is modified.Files which are present in head commit but not in

current folder it comes under deleted files.
It will only allow commit to happen only if one of the array is non empty. If all the array length are 0 then it will not allow to do commit since no changes have been made, its no use of doing commit.If commit occurs then it will display commit details and also append it in .git_log file of current branch. It also changes content of .head.txt file to hash value of new commit.Also for commit it copies current version of all csv files to a folder with name of new hash value in current branch name folder.

3.7 git_checkout

In this case also it stores files in same type of arrays modified files,deleted files and new files.It will only allow to do checkout if length of all the array is zero, otherwise the changes made will be lost.Also it will allow to do checkout if at least one commit is made.
Now submission.sh will call check_out.sh and pass all arguments to it.Now there are three cases :-

1. $1 ==“branch” :-If first argument is branch then its for branch change.during branch change it first remove all csv files and then copy all the csv files from last commit of checkout branch to current folder. It also updates text of .head.txt and .currentbranch.txt.

2. $1 ==“-m” :- if first argument is -m then its for commit checkout using commit message.It first checks if the given message is present or not (All this will be check in git_log file of current branch folder). If not then it prompt “No such message present”. If more than one commit have same message then it prompts conflict in messages.
   In case of only one commit with given message present , then we obtain the Hash_value of corresponding commit message using grep and cut and do the same procedure of remove and copy files and also update of .head.txt and .currentbranch.txt.

3. Else :- commit checkout using Hash_value. Same procedure as that of commit message, just the difference is that in this we will be using Hash value inplace of message.

4 Utilities

The programming languages used in this project by me are:

- Bash[5]
- Sed & Awk
- Basic Python
- Python Libraries

- Numpy[5]
- Matplotlib[1]
- pandas[2]
- tkinter[3]
- sys [4]

References

[1] https://www.w3schools.com/python/matplotlib_intro.asp.

[2] https://pandas.pydata.org/docs/.

[3] https://docs.python.org/3/library/tk.html.

[4] https://docs.python.org/3/library/sys.html.

[5] Prof. Kameshwari Chebrolu. Lecture notes, 2024.