CS 251 Inlab2 - Bash, Git and Markdown

Useful instructions for the Assignment

1. **Remember, you need to change the permission of the .sh file before executing it:**
   chmod u+x <script file> or chmod 777 <script file>

2. **Follow the naming convention of the submission files strictly.**

3. **Remember to follow the exact output formats specified. Extra characters will lead to incorrect evaluation by the autograder.**

4. **Inside the script if you use any temporary file, write a command to remove it after the operation within that script only.**

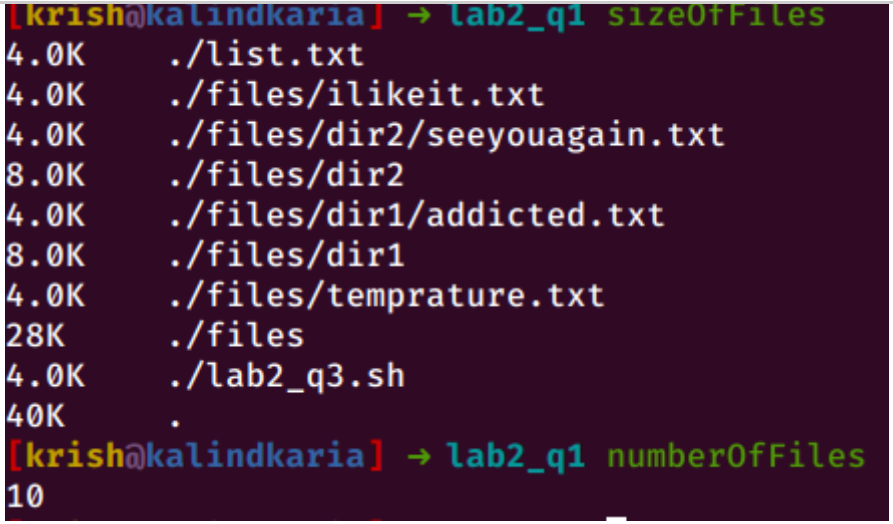
Bash (20 points)

Q1. [10 points=5 + 5]

A sample folder **lab2_q1** is provided in the lab2-resources folder. You need to execute the aliases for this folder.

a) You are required to modify your **.bashrc** file (which is a hidden file in your home folder) and create an alias inside it.

- The alias (***sizeOfFiles***) should be able to get the space usage of all the files and folders and display it on your screen.
- The output of your script should display a list of files and directories in the current directory (should recursively do it for all the files and directories within) along with its size on the Terminal (to be more "precise").
- Output should be in **human readable format** where each file/directory should have its size with its respective units. The output format is not strict apart from this (will be manually graded during viva).
- Look out for a standard bash command which does this for you.



- For Q1 a) and b), commands along with their brief description should be added in the **q5.md** file which is to be submitted in the **Markdown** question (q5) of this inlab.

Q2. [10 points]

A Teacher and some Students are standing in a row. The teacher asks the first student standing in the row to distribute candies on the occasion of Independence day. The teacher always gets zero candy and the first student who distributes the candies will always get one candy. The distributor student should distribute candies such that the i^{th} student (except teacher and distributor student) in row should get candies

(including teacher and student) standing in row should get.

Input: ./candies.sh 5
Output: 0 1 1 2 3
You can run below command and verify the expected output

```
[krish@kalindkaria] → inlab_soln (U!ᚠ main) ./candies.sh 5 | cat -A
0 1 1 2 3 $
[krish@kalindkaria] → inlab_soln (U!ᚠ main)
```

Explanation: If there are 5 persons standing in row,
1. Teacher gets 0 candy
2. Distributor Student gets 1 candy
3. Student at third position gets 1 Candy(0 + 1).
4. Student at fourth position gets 2 Candies(1 + 1).
5. Student at fifth position gets 3 Candies(1 + 2).

Git - GitHub (20 points)

Git is a powerful version control system allowing easy collaboration across developers. **GitHub** is a cloud-based git repository hosting service. It's an online database that allows you to keep track of and share your Git version control projects outside of your local computer / server.

Before moving forward make sure you have signed up on GitHub.
Use this https://training.GitHub.com/downloads/GitHub-git-cheat-sheet.pdf for quick help with the commands.

commits to correct past mistakes shouldn't be there.

Q3. [8 points]

(clone, add, commit, pull, push, gitignore)

The code with final updates is generally put on the main / master branch while different users work on separate branches to implement various features and after finalizing it, they merge it with the main / master branch.
We attempt to do the same and build a C++ string-hashing library. The resources for this task are in the **git/** folder in resources provided on moodle.

1. Login to GitHub and create a **private** repository named **<roll_no>-git** . Initialize it with a README.
2. Clone the repository into your local system.
3. Copy the files from resources to the repository.
4. Add the files **hashing.hpp** (contains declaration of string_hash function) and **main.cpp** (read the file to know what it does) to the staging area and commit with a message **"Add hashing header and string read"** (Copy exact same message).

   git status and **git log** come in handy to analyze the staging area and commit history.
5. Push changes to the main branch in the remote GitHub repository. You can visit the GitHub repo and see the committed changes there.
6. In **hashing.cpp** implement the *int hash_string(string s)* function that sums up the ascii values of the

argument provided to test.sh.
Usage: (output shown for m = 13)
$ ./test.sh weL
output:
10

8. Create a .**gitignore** file and ignore **.out** and **.o** files
9. Test if everything is working as expected and then add (.gitignore, main.cpp, hashing.cpp, test.sh), commit with the commit message **"Add basic hash function implementation"** and push to remote repo.

Copy this repository and rename it to **q3** and add it to the submission folder. Do not tamper with the **.git** folder.

After running **git log --graph --pretty="%s" --all**, the following should be the set of commits visible:

```
* Add basic hash function implementation
* Add hashing header and string read
* Initial commit
```

Q4. [12 points]

(branch, checkout, log, merge)

Suppose you have an idea of a better hash function and decide to implement it in a separate feature branch. Do the following steps:

1. Create a branch **hash_function** and checkout to it.
2. Implement a polynomial rolling hash function by updating the function definition of hash_string(string s)

hashing.hpp.

3. Run the script **test.sh** to check if everything works as expected and then add, commit with the commit message being **“Add polynomial rolling hash function”** and push the branch to upstream .
   For m = 13 and p = 89,
   $ ./test.sh weL
   output:
   0
4. Meanwhile some other user might be working and would have commited code on main. For the sake of learning, mimic this by checking out to the main branch and change the declaration and definition of
   int hash_string(string s) to *int hash_string(string s, int m)*
   And pass **m** as the **smallest odd prime** >= (99 - last 2 digits of your roll no) in main.cpp to hash_string function. Now add (hashing.hpp, hashing.cpp and main.cpp), commit with the commit message **“Update parameters of hash function”** and push to upstream.
   For m = 89,
   $ ./test.sh weL
   output:
   29
5. You can view a tree-like structure of the repo using **git log --graph --all** or use **git log** for viewing commits for the current branch.
   Now you want to merge the two branches.
   Checkout to hash_function branch. And initiate a merge with main.
   There will be merge conflicts which you need to resolve by keeping the new implementation of hash

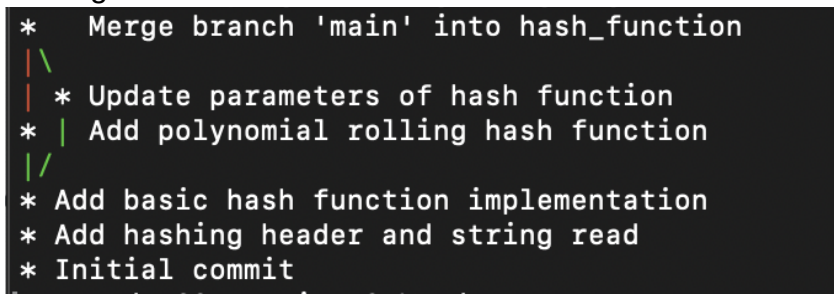
output:
30

Now observe the log graph. You can notice that the commits in the main branch are visible in the hash_function branch and there is a new merge commit too.

6. Now we would like to have the changes fast forwarded in the main branch also. For this, checkout to main and merge with hash_function. Now push to the remote repo.

Copy this repository and rename it to **q4** and add it to the submission folder.

After running **git log --graph --pretty="%s" --all**, the following should be the set of commits on various branches:



Markdown

Q5. [5 points]

Your task is to create an elegant README.md for the repository that you have built in the git task. Just go to your GitHub repo and click on the pencil icon on top right of README.md and start editing.
- https://www.markdownguide.org/cheat-sheet/ use this as reference

The README should have following components :

- Your name and Rollno (put it in a tabular format)
- What is the repository about (some description of the hash function) and its utility (can you think of any ?) ?
- Instructions to run the code
- Your solution to q1 of bash
- All the references you have used in this lab (the exact links; apart from what have been linked in the PS or have been provided)

Use different levels of headings, table, math equation, formatted commands, bullets and bold important words. (0.5 marks for each of these and 2 marks for the content)

Commit this under the default message "**Update README.md**" and show it to the TA during your viva (keep the name README.md in the repository).
Rename it to **q5.md** and add it in your submission (just this file).

Submission Instructions

Compress it into a tarball using the following command

tar -cvzf <roll_no>_inlab2.tar.gz <roll_no>_inlab2

```
<roll_no>_inlab2
├── candies.sh
├── q3
│   ├── .git/
│   ├── .gitignore
│   ├── README.md
│   ├── a.out (optional)
│   ├── hashing.cpp
│   ├── hashing.hpp
│   ├── main.cpp
│   └── test.sh
├── q4
│   ├── .git/
│   ├── .gitignore
│   ├── README.md
│   ├── a.out (optional)
│   ├── hashing.cpp
│   ├── hashing.hpp
│   ├── main.cpp
│   └── test.sh
└── q5.md
```

Note: README.md files in q3 and q4 directory are the default README that are created when a new GitHub repository is set-up and contains only the repository name.

CS 251 Inlab2 - Bash, Git and Markdown