As in the git lab, you have been provided with 3 scripts in the scripts folder. To execute any, navigate to the scripts folder and run the required script with “bash <script\_name>.sh”. Be sure to navigate to the scripts directory before running the script.

reset.sh :- Used to reset everything in the repository to scratch. Execute this script to start the activity.

submit.sh :- Run the submit.sh script before evaluating or submitting/committing your directory in VLab

reopen.sh :- After submitting, if you wish to start off from where you left, run this script to set up the directory as it was when you submitted it.

\*\*\*CAUTION :- USE RESET.SH VERY CAREFULLY! IT CAN ERASE ALL YOUR PROGRESS!\*\*\*

After running reset.sh, go to the labDirectory and enter the repo folder. Now the story begins...

A friend of yours tried to merge two branches (master branch, where the HEAD was, and the development branch) of a git repository but couldn’t resolve the merge conflicts in a file. Unfortunately they got too frustrated trying to resolve the errors and finally created a new repository with only a single file (containing the merge conflict headers added by Git). They bring this new repo to you, and now ask you to help them.

You are given a git repository with a single branch (master) and a single commit, containing a file (file.cpp) which includes within itself the merge conflict headers added by git. Your aim is to recreate the original 2 files from this file using the merge conflict headers.

After recreating the files, name the file that is identical to the file originally in the master branch (of the original repo) as “file\_master.cpp” and name the file that is identical to the file in the development branch (of the original repo) as “file\_development.cpp”. The original file.cpp should not be present anymore.

Make another commit (that contains only the 2 newly created files) with the message “Recreated files”. (Make sure that this part of the task is done in only one commit)

Now, from this commit, create a branch named “development” and in this branch, make a commit with “file\_master.cpp” removed, with the commit message - “Removed file\_master.cpp”.

In the original master branch, make a commit with “file\_development.cpp” removed, with the commit message - “Removed file\_development.cpp”.

The correctness of the files will be judged on the functionality of the code, so don’t worry about matching the original file exactly. As long as the code works as expected, you will be awarded full marks.

“Problem here” - - - -> “Recreated Files” - - - - > “Removed file\_development.cpp” (master)

\

- - - -> “Removed file\_master.cpp” (development)

The above is a basic structure of what the final commit history should look like.

Effectively, this exercise makes a repository similar to what your friend had accidentally deleted.

For checking the recreated files, you may use the following test cases :-

File\_master.cpp

Input :-

4

4 2 1 3

5

1 3

2 4

1 4

3 4

2 2

Output :-

7

6

10

4

2

For file\_development.cpp :-

Input :-

4

4 2 1 3

5

1 3

2 4

1 4

3 4

2 2

Output :-

8

6

24

3

2

Grading -

The given evaluation scripts check only for the repository structure, commit messages, etc. and not if the recreated files are running properly. This will be done on the server and the results hidden from you. It is not necessary that if you get full here, you will also get full in the full activity.