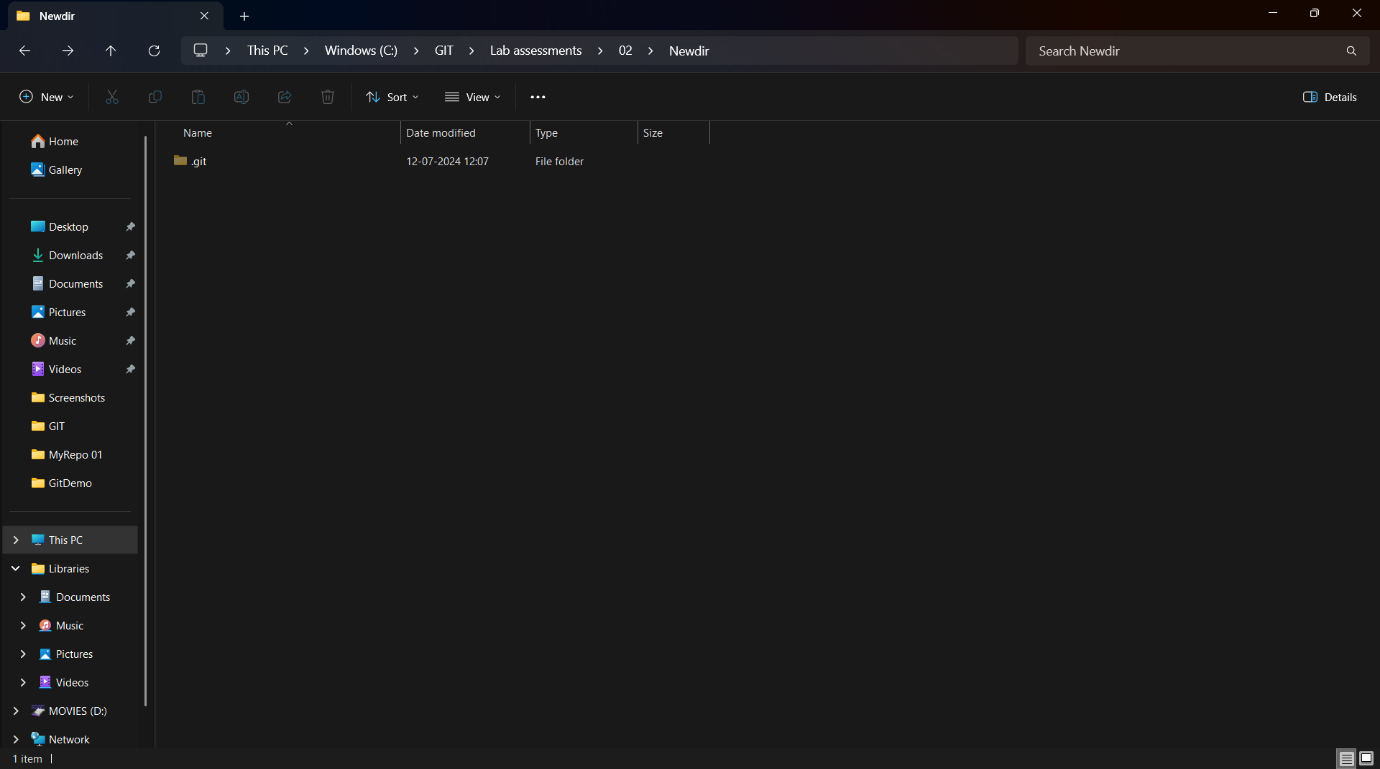
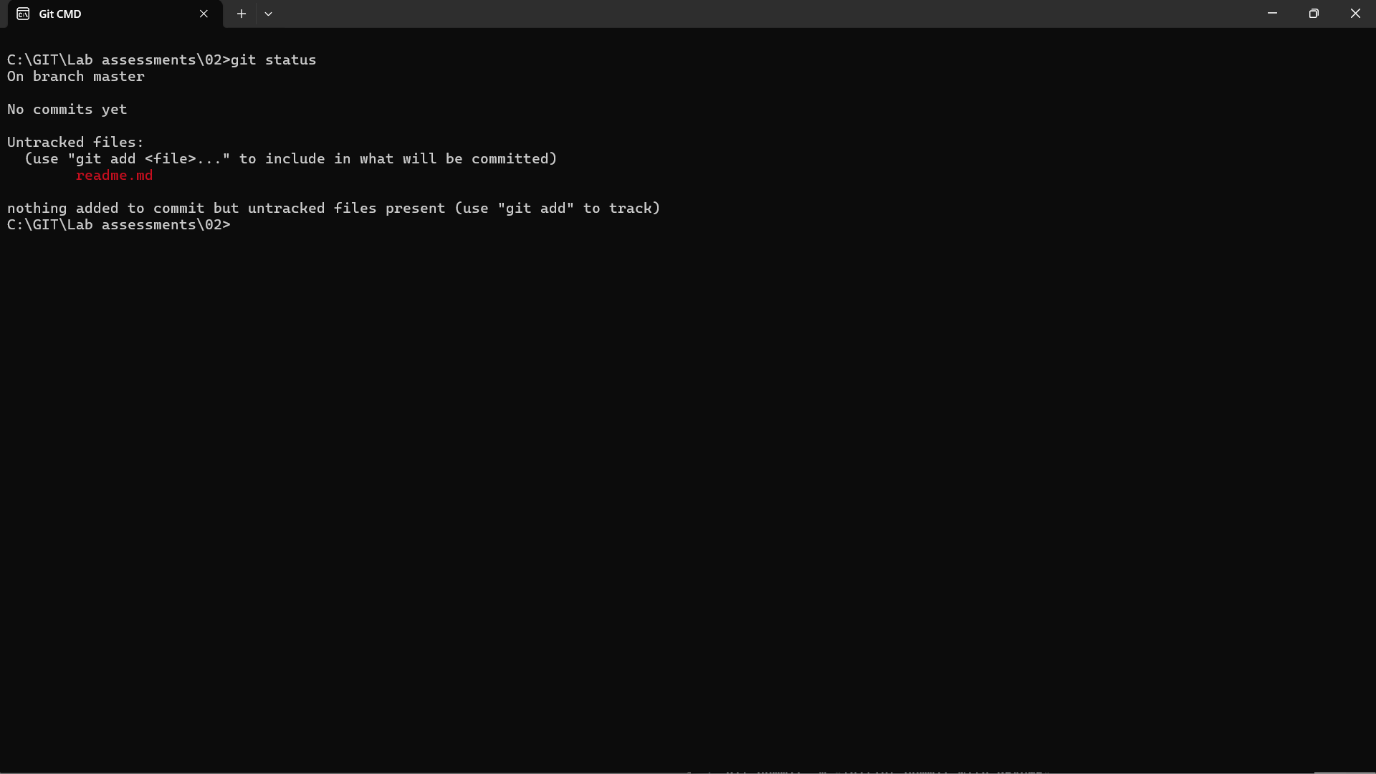
Name: Arunkumar K (73152113010) Email: [arunkumarkcse2022@ksrce.ac.in](mailto:arunkumarkcse2022@ksrce.ac.in)

1.Use the init command to create a Git repository in that directory and Observe that there is now a .git directory.

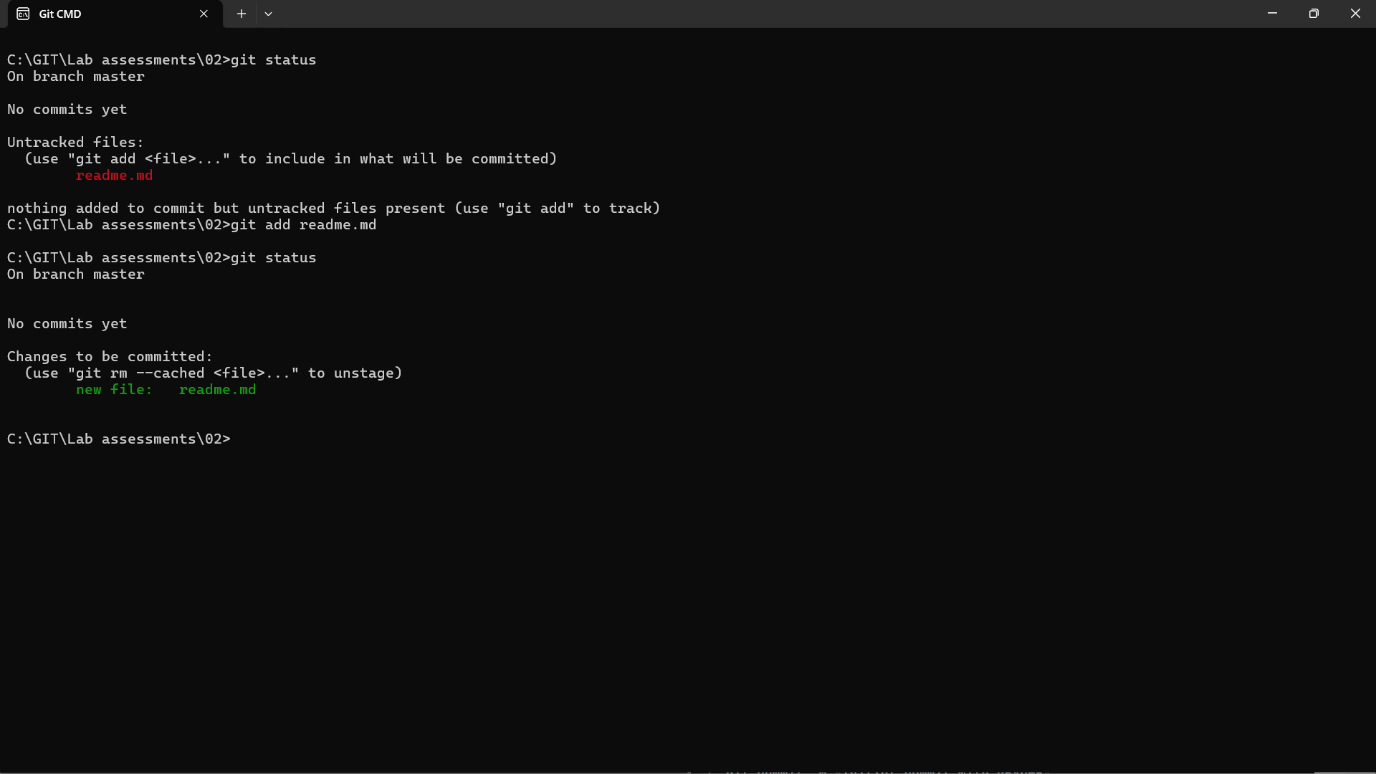


2. Create a README file and look at the output of the status command, the README you created should appear as an untracked file

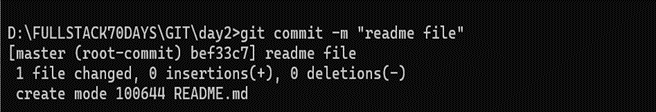


3. Use the add command to add the new file to the staging area. Again, look at the output of

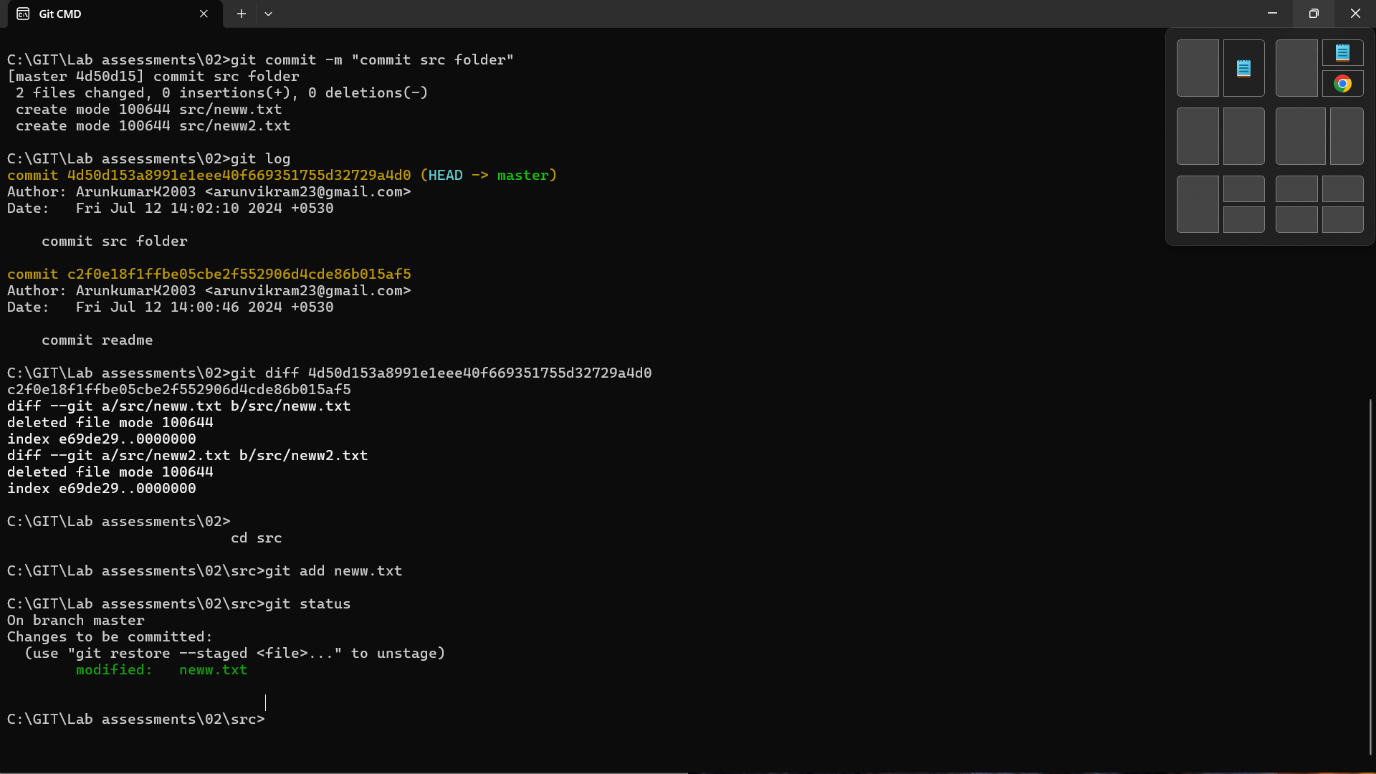
the status command.



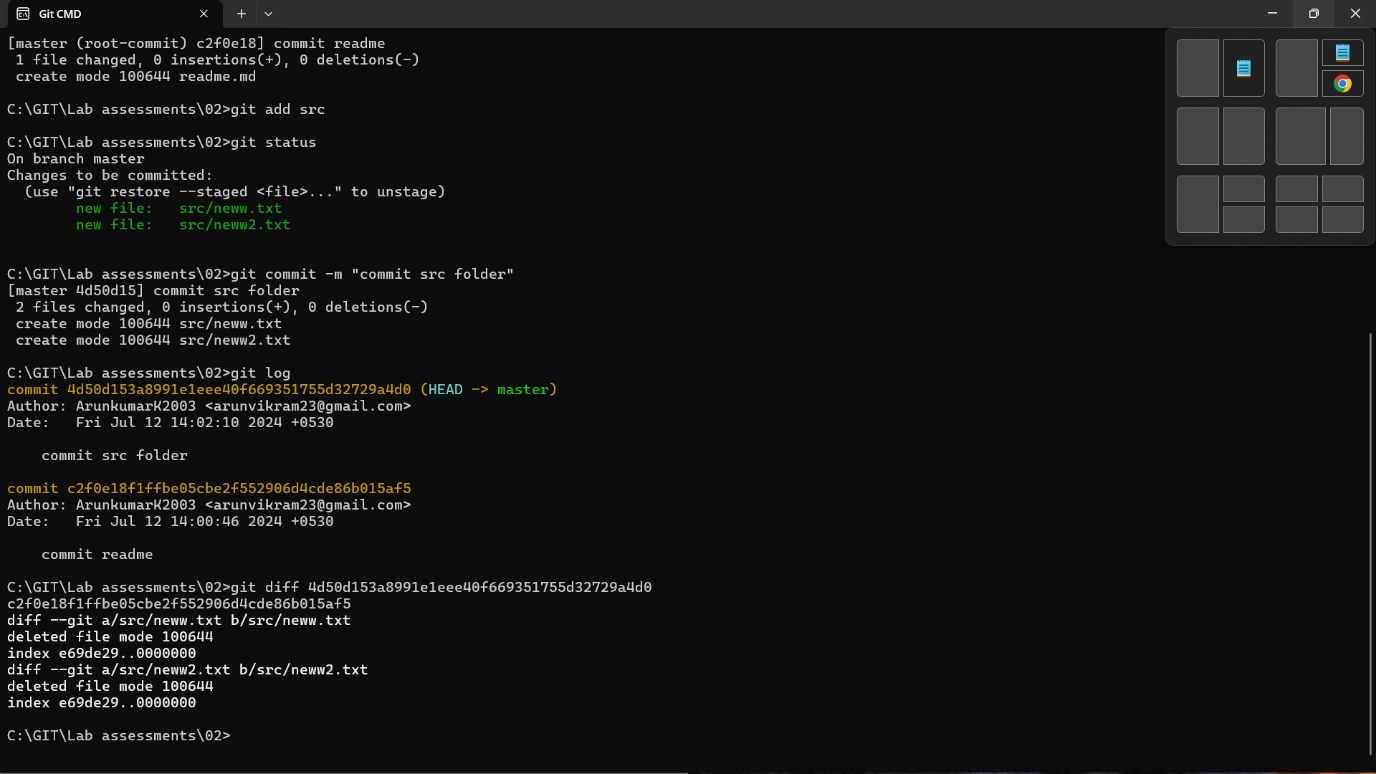
4. Now use the commit command to commit the contents of the staging area.



5. Create a src directory and add a couple of files to it. Use the add command, but name the directory, not the individual files. Use the status command. See how both files have been staged. Commit them.

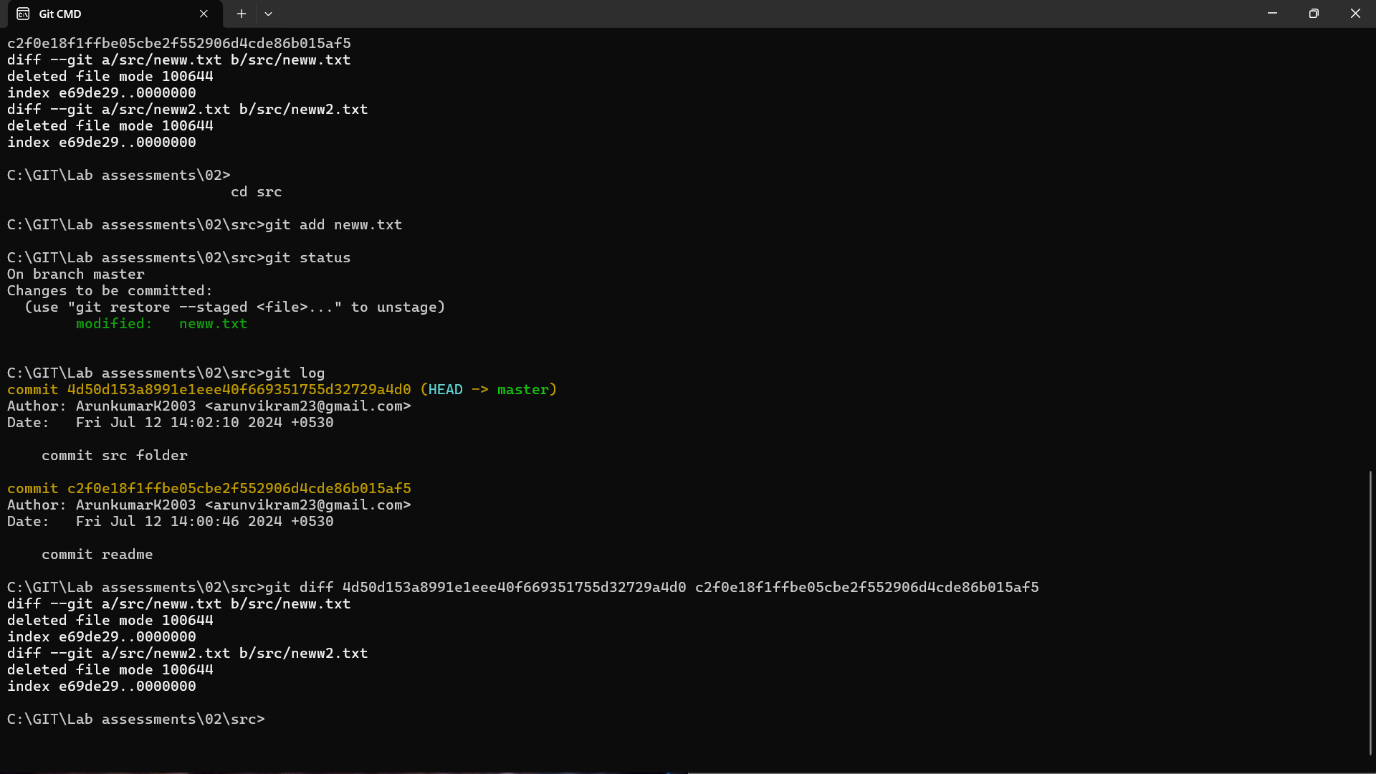


6. Make a change to one of the files. Use the diff command to view the details of the change.



6. Next, add the changed file, and notice how it moves to the staging area in the status output. Also observe

that the diff command you did before using add now gives no output. Why not? What do you have to do to see a diff of the things in the staging area?



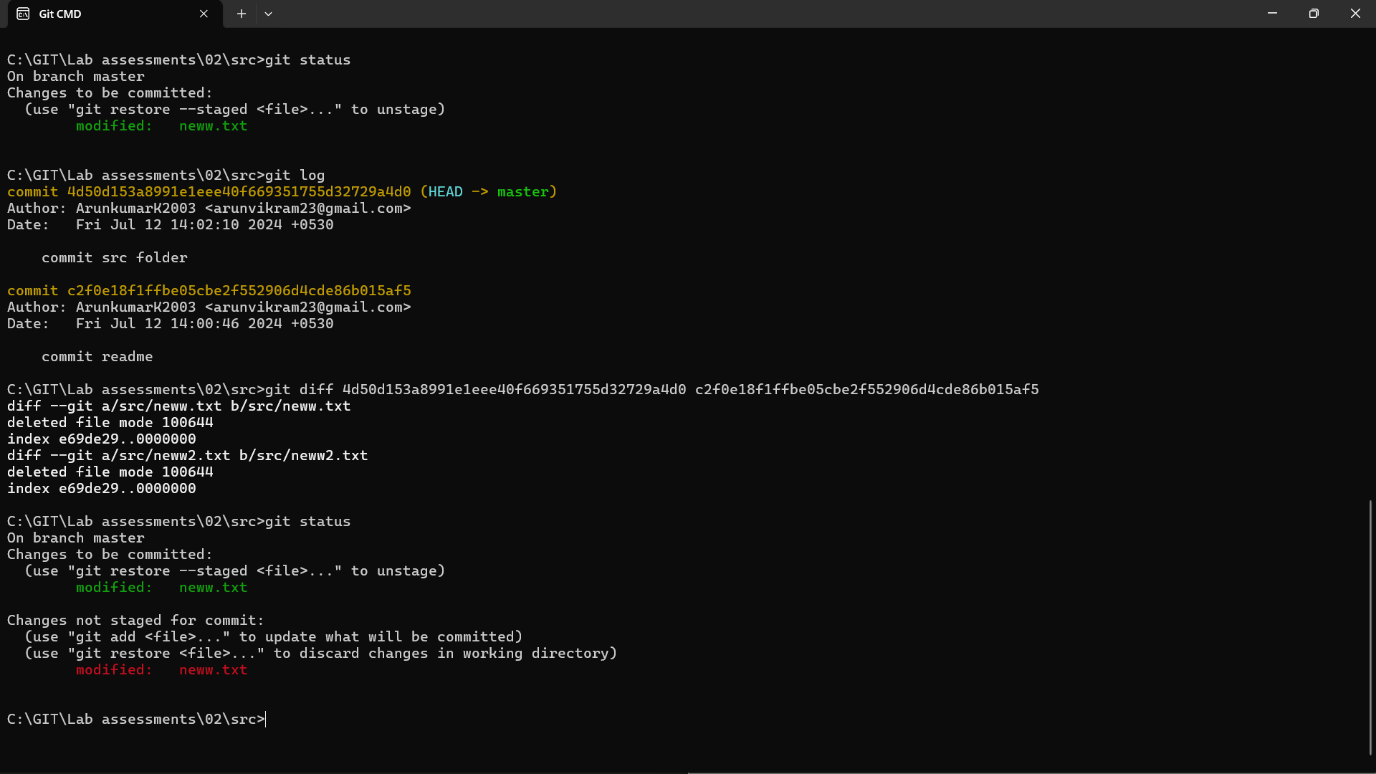
7. Now – without committing – make another change to the same file you changed in step 10.

Look at the status output, and the diff output. Notice how you can have both staged and

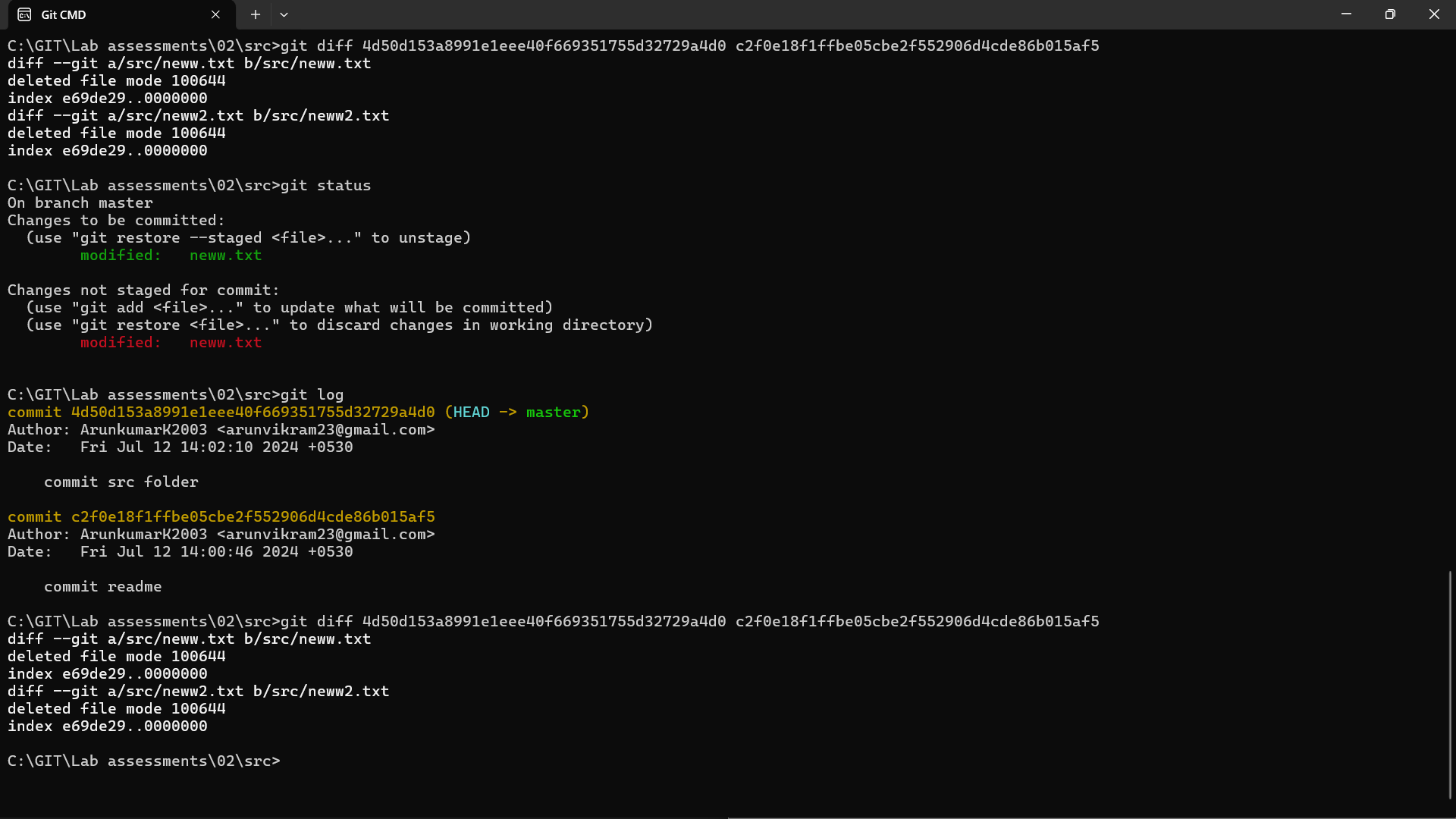
unstaged changes, even when you’re talking about a single file. Observe the difference when

you use the add command to stage the latest round of changes. Finally, commit them. You

should now have started to get a feel for the staging area.

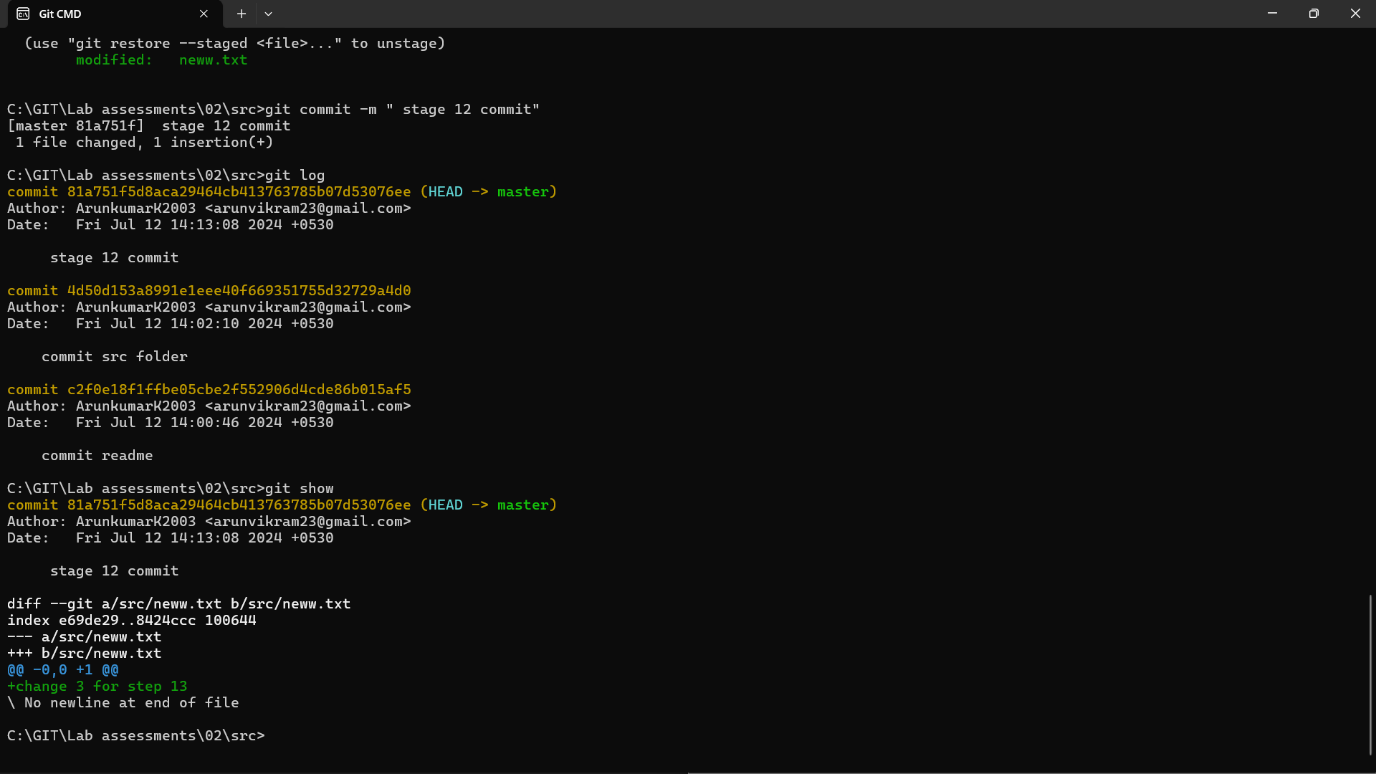


8. Use the log command in order to see all of the commits you made so far.



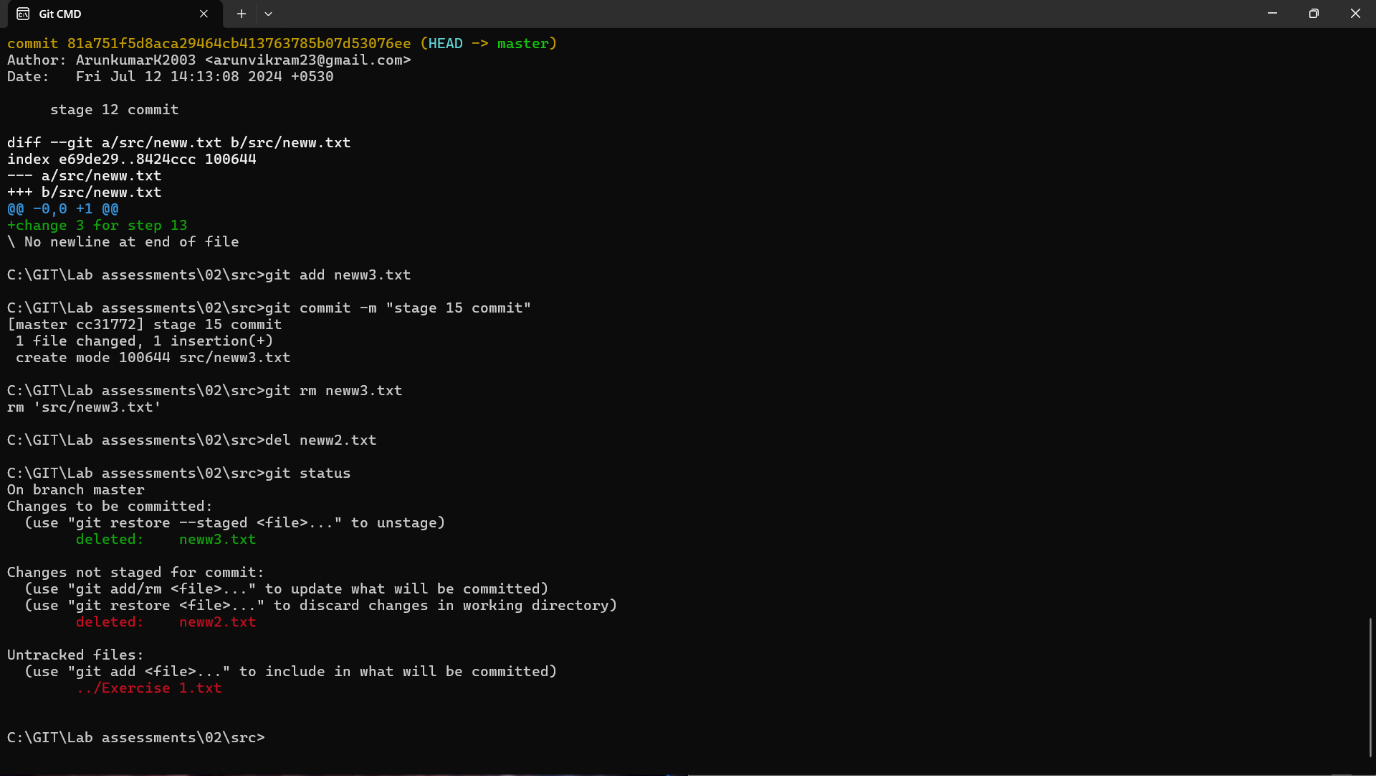
9. Use the show command to look at an individual commit. How many characters of the

commit identifier can you get away with typing at a minimum?



10. Make a couple more commits, at least one of which should add an extra file.

Stretch Task

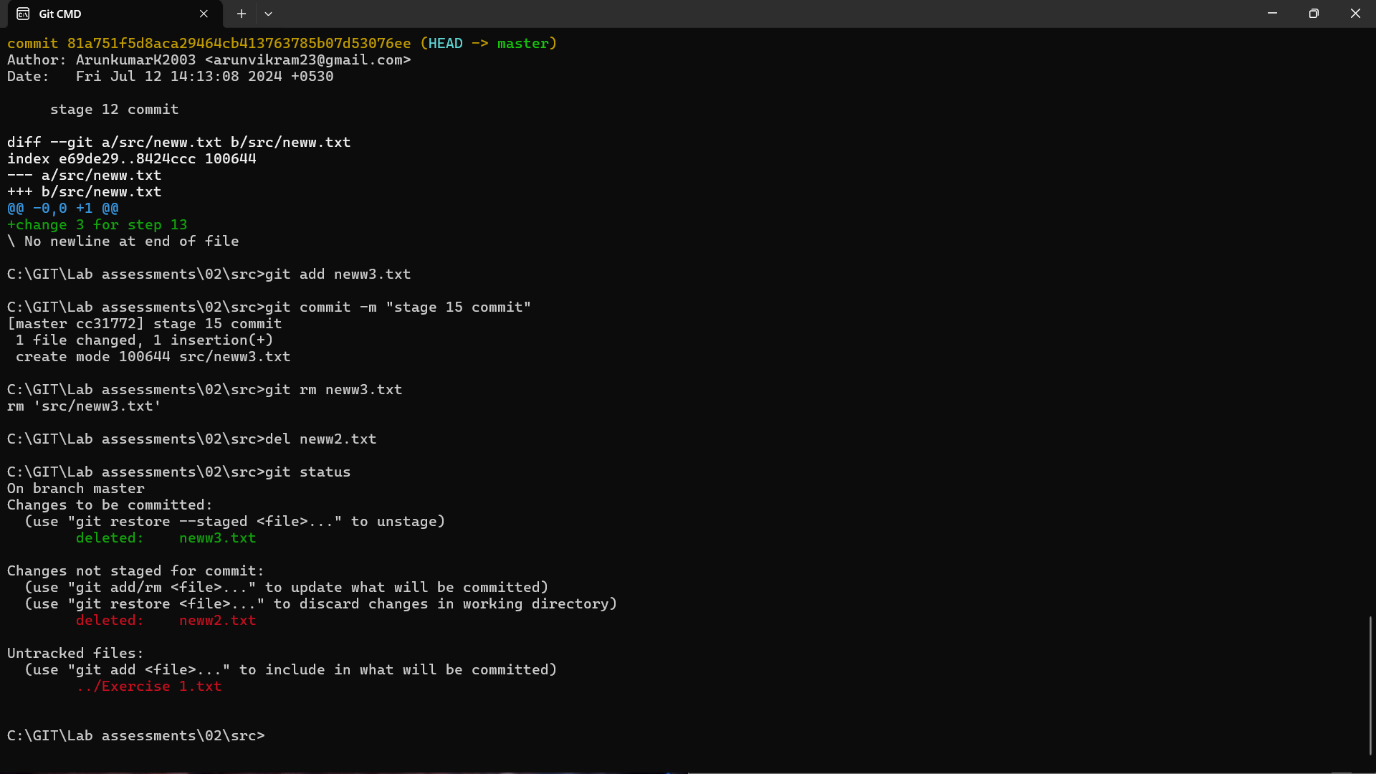


***Stretch Task***

1. Use the Git rm command to remove a file. Look at the status afterwards. Now commit the

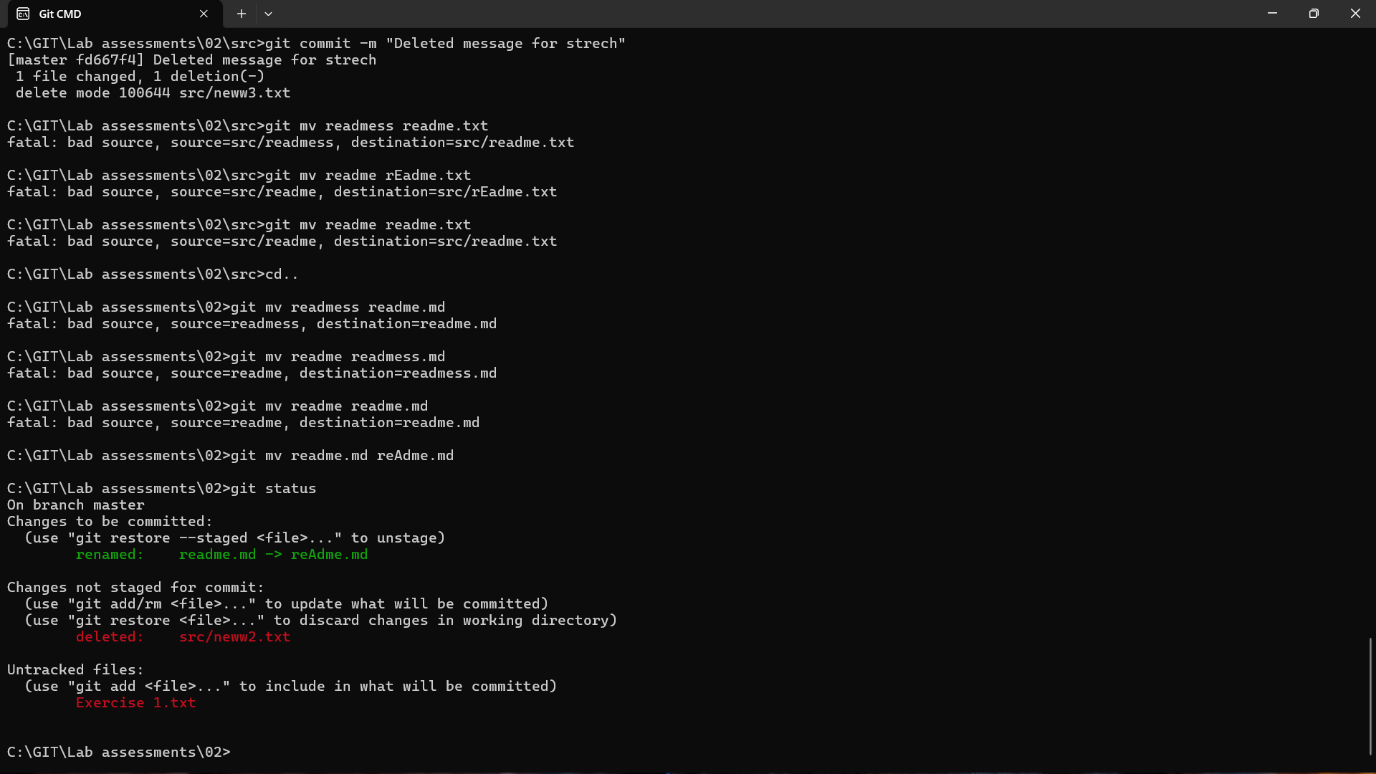
Deletion and delete another file, but this time do not use Git to do it; e.g. if you are on Linux, just use the

normal (non-Git) rm command; on Windows use del and also verify the status of the file.



2. Use the Git mv command to move or rename a file; for example, rename README to

README.txt. Look at the status. Commit the change.



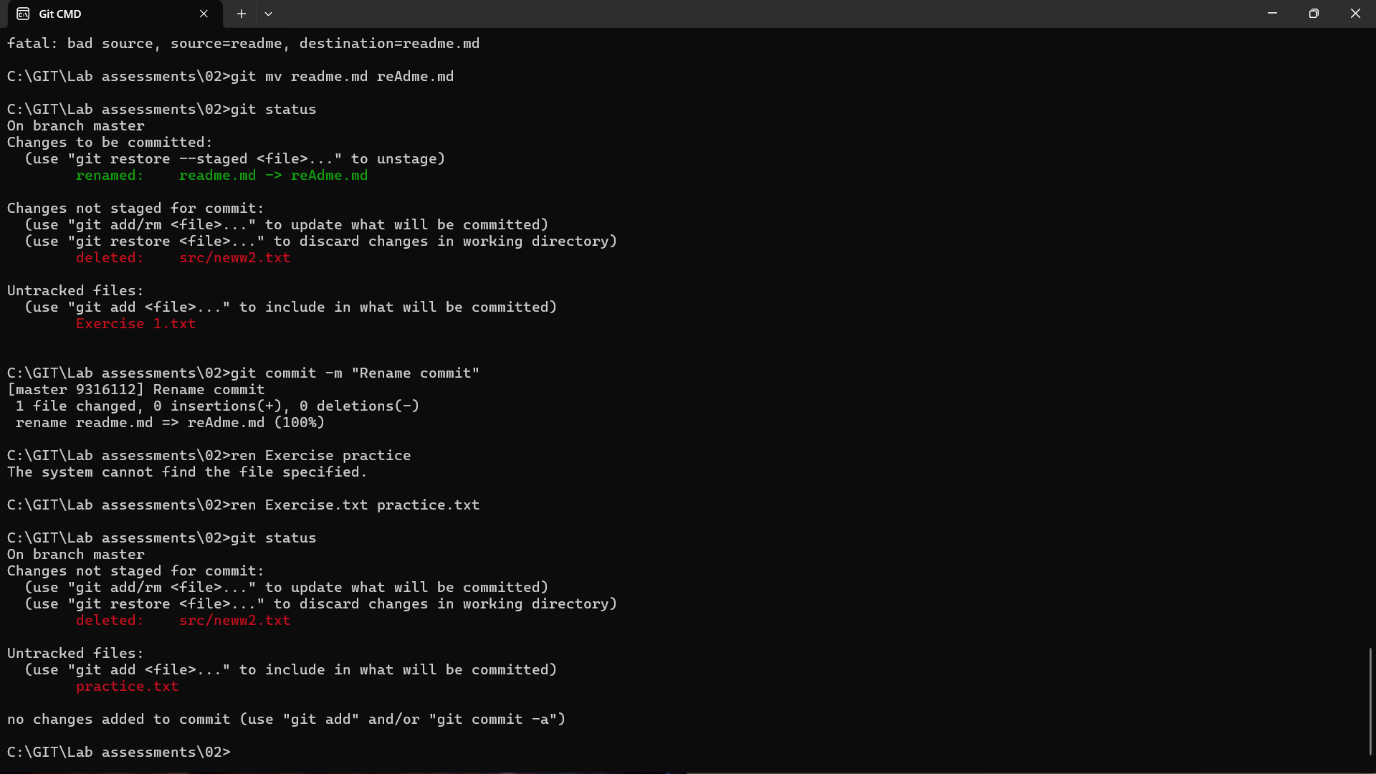
3. Now do another rename, but this time using the operating system’s command to do so. How

does the status look? Will you get the right outcome if you were to commit at this point?

(Answer: almost certainly not, so don’t. ) Work out how to get the status to show that it

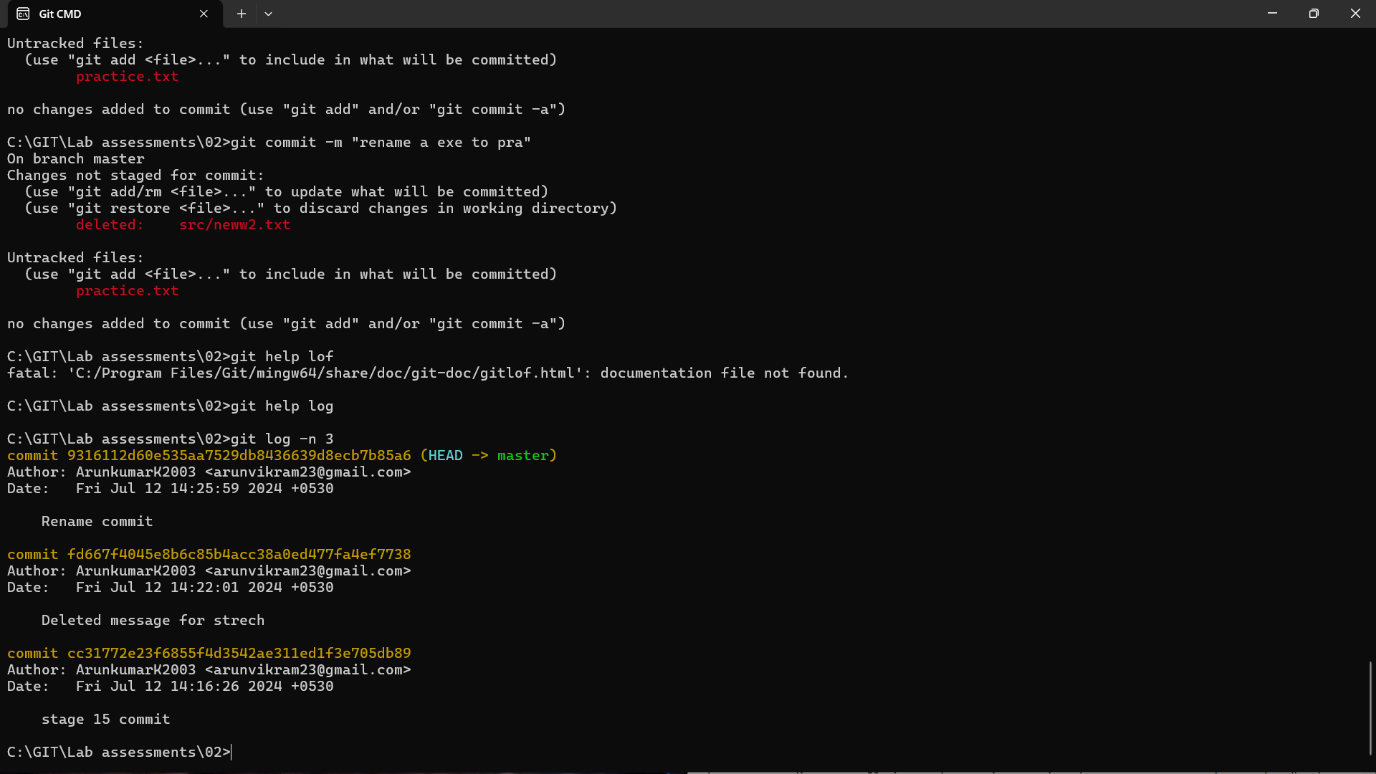
will not lose the file, and then commit. Did Git at any point work out that you had done a

rename?



4. Use git help log to find out how to get Git to display just the most recent 3 commits. Try

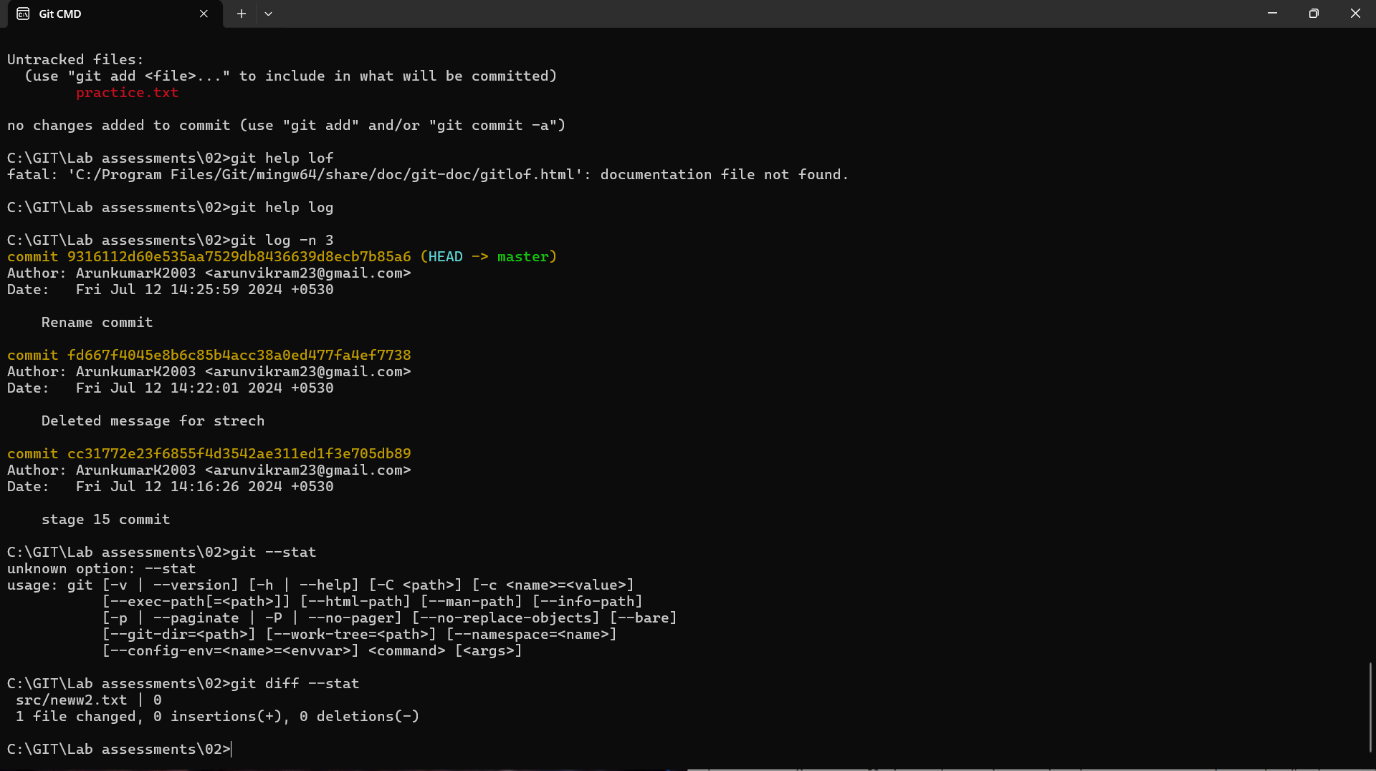
it.



5. If you don’t remember, look back in the slides to see what the --stat option did on the

diff command. Find out if this also works with the show command. How about the log

command?



6. Imagine you want to see a diff that summarizes all that happened between two commit

identifiers. Use the diff command, specifying two commit identifiers joined by two dots

(that is, something like abc123..def456). Check the output is what you expect.

