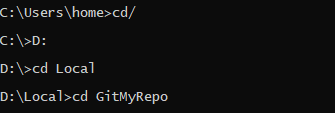
EXERCISE 1

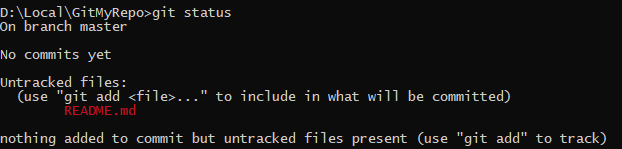
1. Create a new directory and change into it



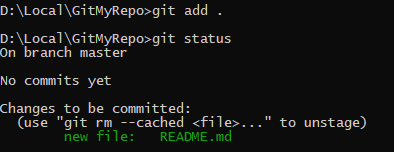
1. Use the init command to create a Git repository in that directory & 3) Observe that there is now a .git directory.



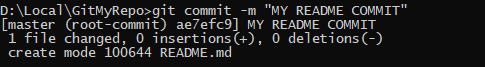
4)Create a README file. & 5) Look at the output of the status command; the README you created should appear as an untracked file.



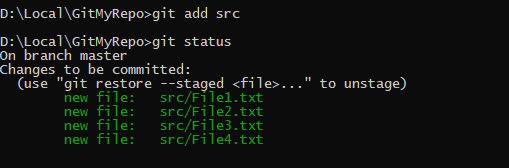
6) Use the add command to add the new file to the staging area. Again, look at the output of the status command.



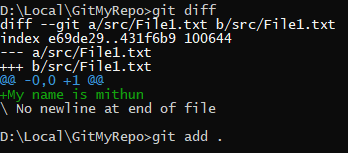
7) Now use the commit command to commit the contents of the staging area.



8) Create a src directory and add a couple of files to it & 9) 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.

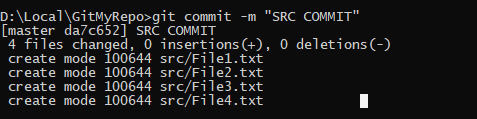


10) Make a change to one of the files. Use the diff command to view the details of the change.

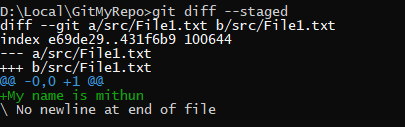


11) Next, add the changed file, and notice how it moves to the staging area in the status output. Also observe that the diff command you did before using add now gives no output.Why not? What do you have to do to see a diff of the things in the staging area? (Hint:

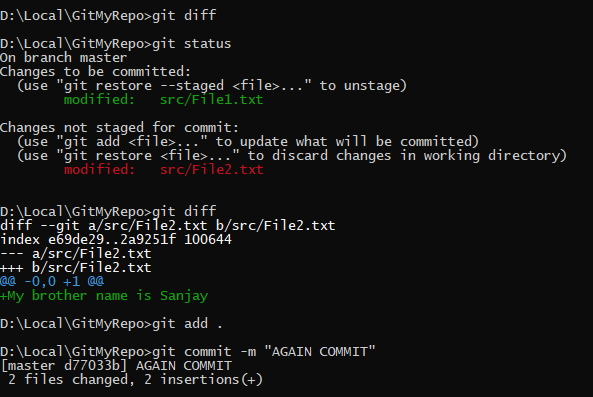
review the slides if you can’t remember.)



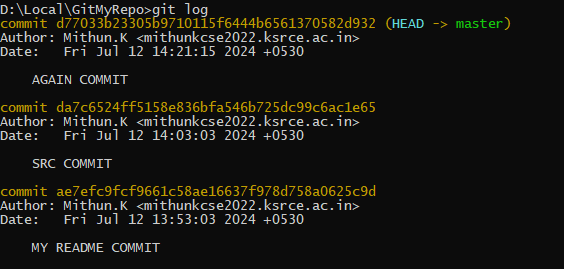
12) Now – without committing – make another change to the same file you changed in step 10.Look at the status output, and the diff output. Notice how you can have both staged andunstaged changes, even when you’re talking about a single file. Observe the difference whenyou use the add command to stage the latest round of changes. Finally, commit them. Youshould now have started to get a feel for the staging area.



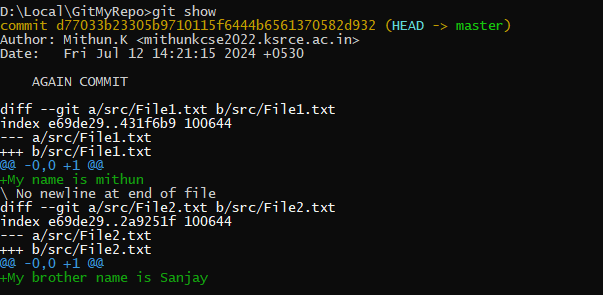
13) Use the log command in order to see all of the commits you made so far

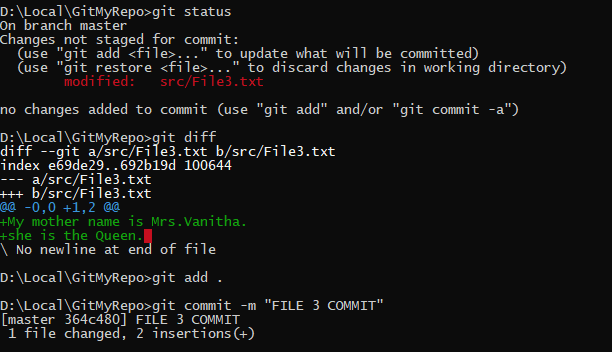


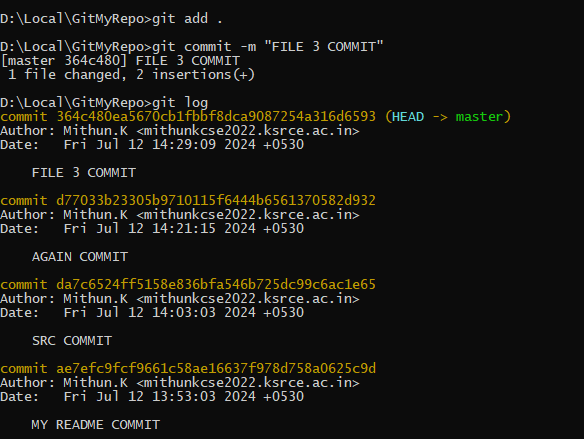
14) Use the show command to look at an individual commit. How many characters of thecommit identifier can you get away with typing at a minimum?



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

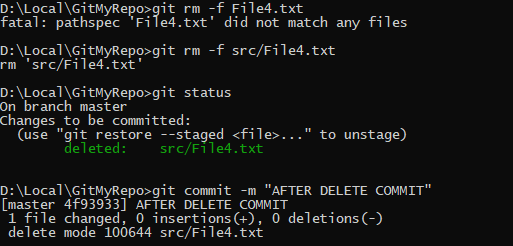




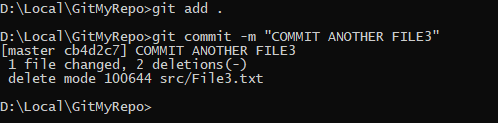


STRETCH TASK

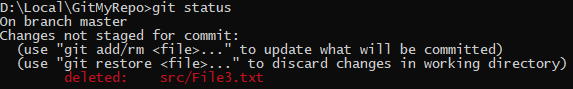
1)Use the Git rm command to remove a file. Look at the status afterwards. Now commit the deletion.



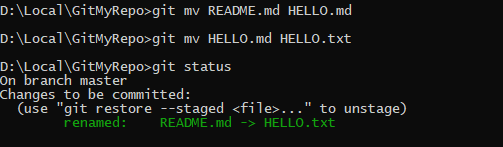
2)Delete another file, but this time do not use Git to do it; e.g. if you are on Linux, just use thenormal (non-Git) rm command; on Windows use del.



3)Look at the status. Compare it to the status output you had after using the Git built-in rm command. Is anything different? After this, commit the deletion.

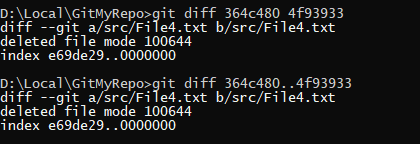


4)Use the Git mv command to move or rename a file; for example, rename README to README.txt. Look at the status. Commit the change.



5) 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?



7) 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.