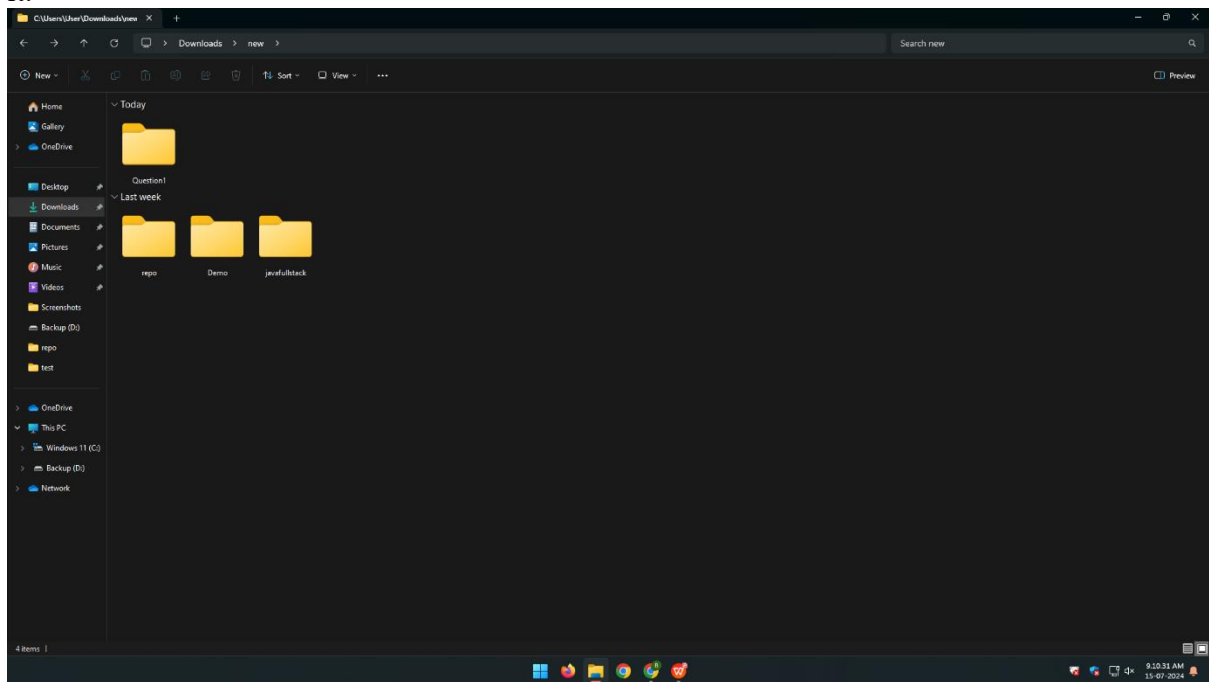


## Exercise 1

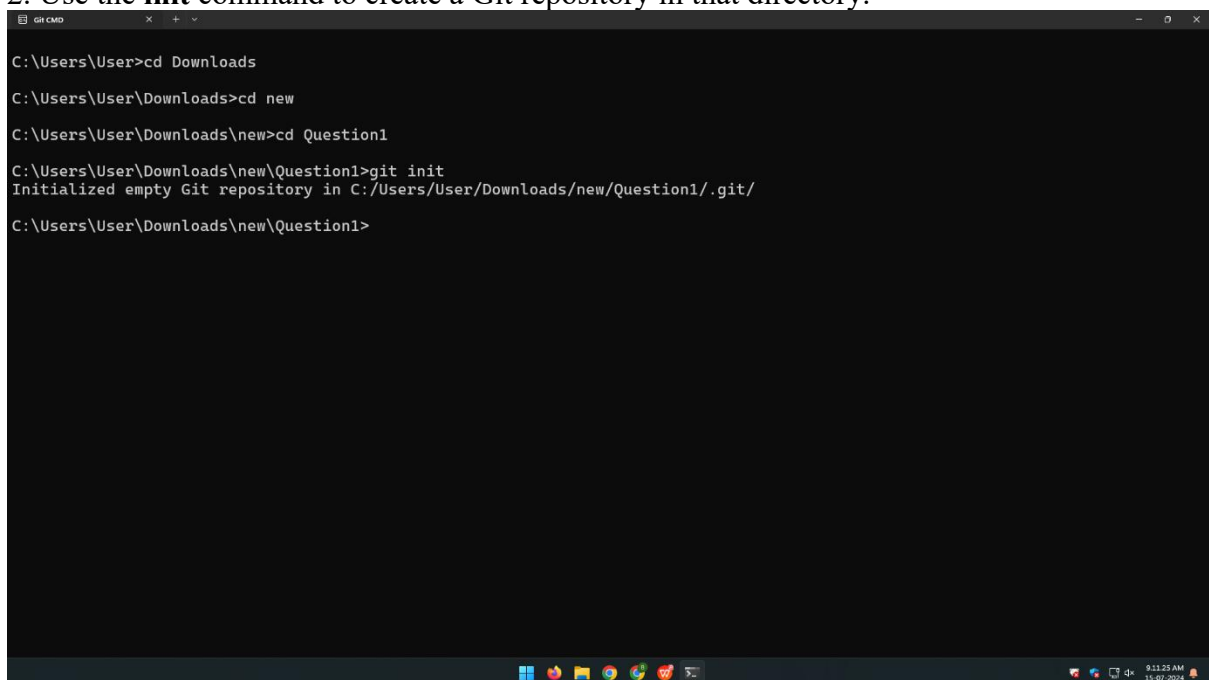
### Main Task

1. Create a new directory and change into it.

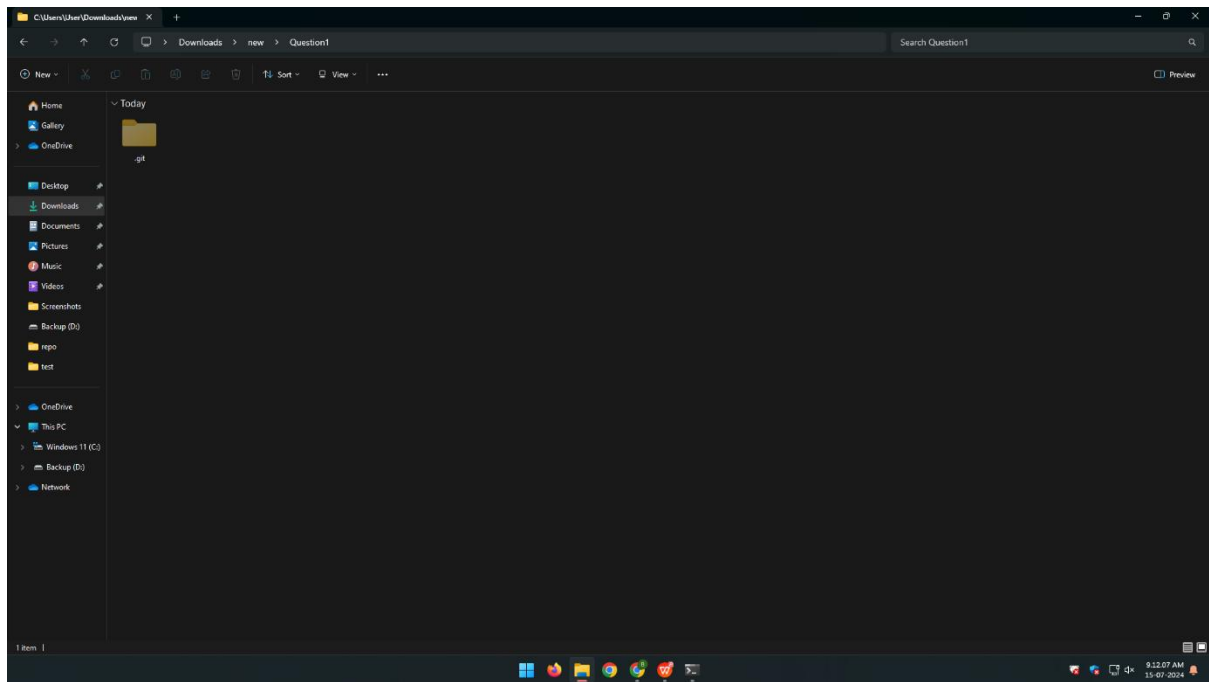


(Qusetion1 is my new Directory)

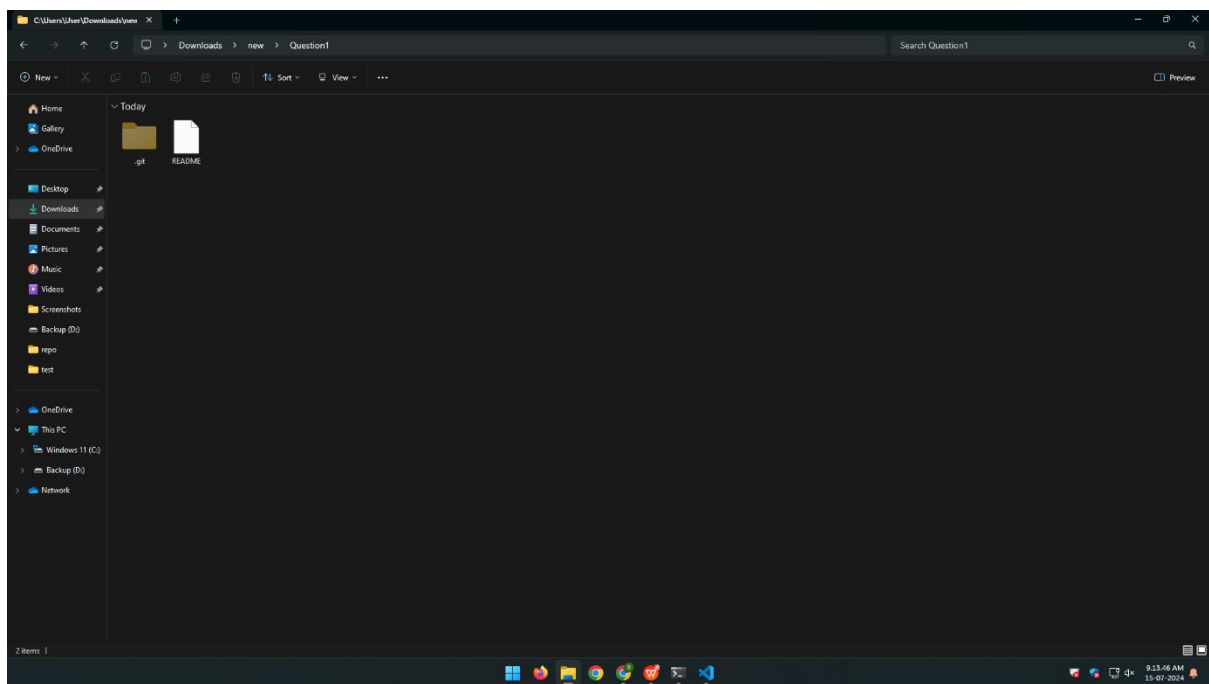
2. Use the **init** command to create a Git repository in that directory.



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.

```
Git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git init
Initialized empty Git repository in C:/Users/User/Downloads/new/Question1/.git/
C:\Users\User\Downloads\new\Question1>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Downloads\new\Question1>
```

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

```
Git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git init
Initialized empty Git repository in C:/Users/User/Downloads/new/Question1/.git/
C:\Users\User\Downloads\new\Question1>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Downloads\new\Question1>git add README
C:\Users\User\Downloads\new\Question1>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README

C:\Users\User\Downloads\new\Question1>
```

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

```
git CMD
C:\Users\User\Downloads\new\Question1>git init
Initialized empty Git repository in C:/Users/User/Downloads/new/Question1/.git/

C:\Users\User\Downloads\new\Question1>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  README

nothing added to commit but untracked files present (use "git add" to track)

C:\Users\User\Downloads\new\Question1>git add README

C:\Users\User\Downloads\new\Question1>git status
On branch master

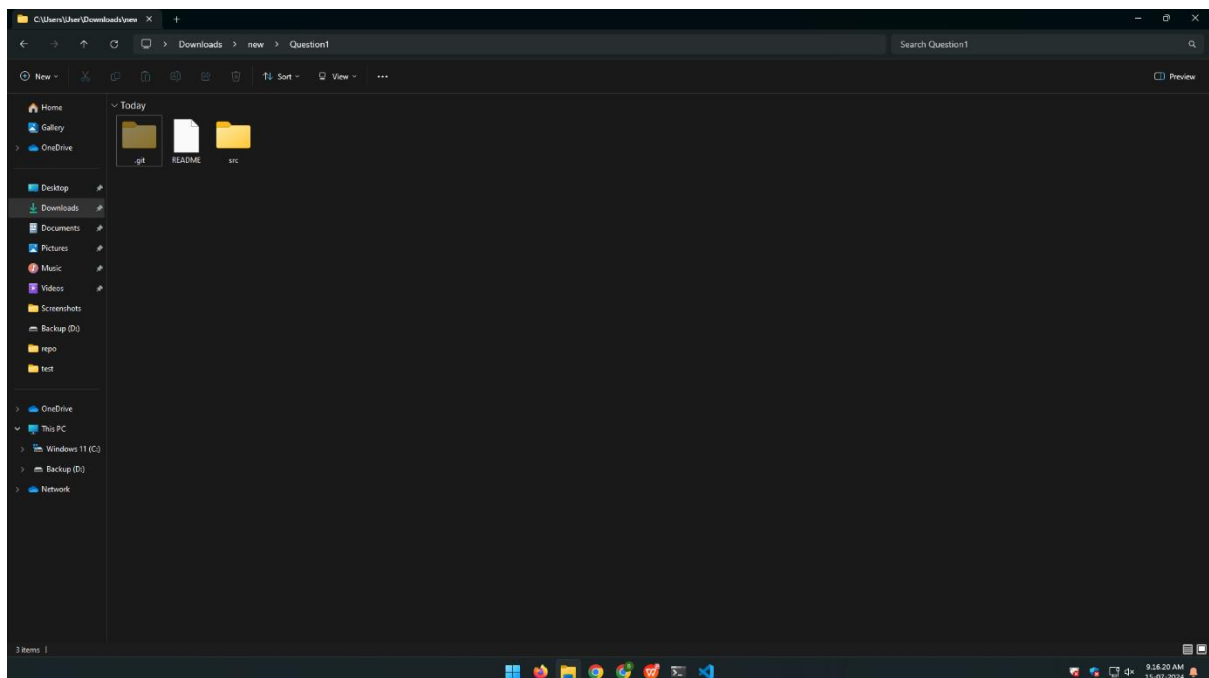
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   README

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit"
[master (root-commit) 8808b09] mycommit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README

C:\Users\User\Downloads\new\Question1>
```

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.

```
git CMD
nothing added to commit but untracked files present (use "git add" to track)
C:\Users\User\Downloads\new\Question1>git add README
C:\Users\User\Downloads\new\Question1>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit"
[master (root-commit) 8808b09] mycommit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README

C:\Users\User\Downloads\new\Question1>mkdir src
C:\Users\User\Downloads\new\Question1>git add src
C:\Users\User\Downloads\new\Question1>git status
On branch master

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/demo.java
        new file:   src/demo.txt
        new file:   src/example.c

C:\Users\User\Downloads\new\Question1>
```

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

```
git CMD
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit"
[master (root-commit) 8808b09] mycommit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README

C:\Users\User\Downloads\new\Question1>mkdir src
C:\Users\User\Downloads\new\Question1>git add src
C:\Users\User\Downloads\new\Question1>git status
On branch master

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/demo.java
        new file:   src/demo.txt
        new file:   src/example.c

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit2"
[master 946cc27] mycommit2
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/demo.java
create mode 100644 src/demo.txt
create mode 100644 src/example.c

C:\Users\User\Downloads\new\Question1>
```

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.)

```
git CMD
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README

C:\Users\User\Downloads\new\Question1>mkdir src

C:\Users\User\Downloads\new\Question1>git add src

C:\Users\User\Downloads\new\Question1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/demo.java
        new file:   src/demo.txt
        new file:   src/example.c

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit2"
[master 946cc27] mycommit2
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/demo.java
create mode 100644 src/demo.txt
create mode 100644 src/example.c

C:\Users\User\Downloads\new\Question1>git diff
diff --git a/src/demo.txt b/src/demo.txt
index e69de29..41379c0 100644
--- a/src/demo.txt
+++ b/src/demo.txt
@@ -0,0 +1 @@
+bharani
\ No newline at end of file

C:\Users\User\Downloads\new\Question1>
```

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 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.

```
git CMD
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README

C:\Users\User\Downloads\new\Question1>mkdir src

C:\Users\User\Downloads\new\Question1>git add src

C:\Users\User\Downloads\new\Question1>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/demo.java
        new file:   src/demo.txt
        new file:   src/example.c

C:\Users\User\Downloads\new\Question1>git commit -m "mycommit2"
[master 946cc27] mycommit2
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/demo.java
create mode 100644 src/demo.txt
create mode 100644 src/example.c

C:\Users\User\Downloads\new\Question1>git diff
diff --git a/src/demo.txt b/src/demo.txt
index e69de29..41379c0 100644
--- a/src/demo.txt
+++ b/src/demo.txt
@@ -0,0 +1 @@
+bharani
\ No newline at end of file

C:\Users\User\Downloads\new\Question1>
```

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

```
git CMD
+++ b/src/demo.txt
@@ -0,0 +1 @@
+bharani
\ No newline at end of file

C:\Users\User\Downloads\new\Question1>git add .

C:\Users\User\Downloads\new\Question1>git commit -m "my commit 3"
[master 08120b3] my commit 3
1 file changed, 1 insertion(+)

C:\Users\User\Downloads\new\Question1>git diff

C:\Users\User\Downloads\new\Question1>git log
commit 08120b31a9795b534082b6a1587589477050ce2c (HEAD -> master)
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:20:24 2024 +0530

    my commit 3

commit 946cc271da8214878aee4b34d8bc5efefac1d22d
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:18:11 2024 +0530

    mycommit2

commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530

    mycommit

C:\Users\User\Downloads\new\Question1>
```

14. 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?

```
git CMD

    alter the sat.txt

commit c404f13ed4aff6b67bd90c9cc8f2815de4ff5d18
Merge: 2f7a467 e82da02
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:37:05 2024 +0530

    Merge branch 'class'

commit 2f7a467d2cf3df048c1016bc89608c68be9898dc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:36:56 2024 +0530

    the sat.txt

commit e82da02eaf66665d8daf8267e0c877f1c4447dd2
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:35:32 2024 +0530

    the sat.txt

C:\Users\User\Downloads\new\Question1>git show 2f7a467d
commit 2f7a467d2cf3df048c1016bc89608c68be9898dc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:36:56 2024 +0530

    the sat.txt

diff --git a/src/sat.txt b/src/sat.txt
index e69de29..41379c0 100644
--- a/src/sat.txt
+++ b/src/sat.txt
@@ -0,0 +1 @@
+bharani
\ No newline at end of file

C:\Users\User\Downloads\new\Question1>
```

14)How many characters of the commit identifier can you get away with typing at a minimum?

In general, the first 4-7 characters of the commit hash are sufficient to uniquely identify a commit in repositories

For example:

`Git show 2f7a467d`

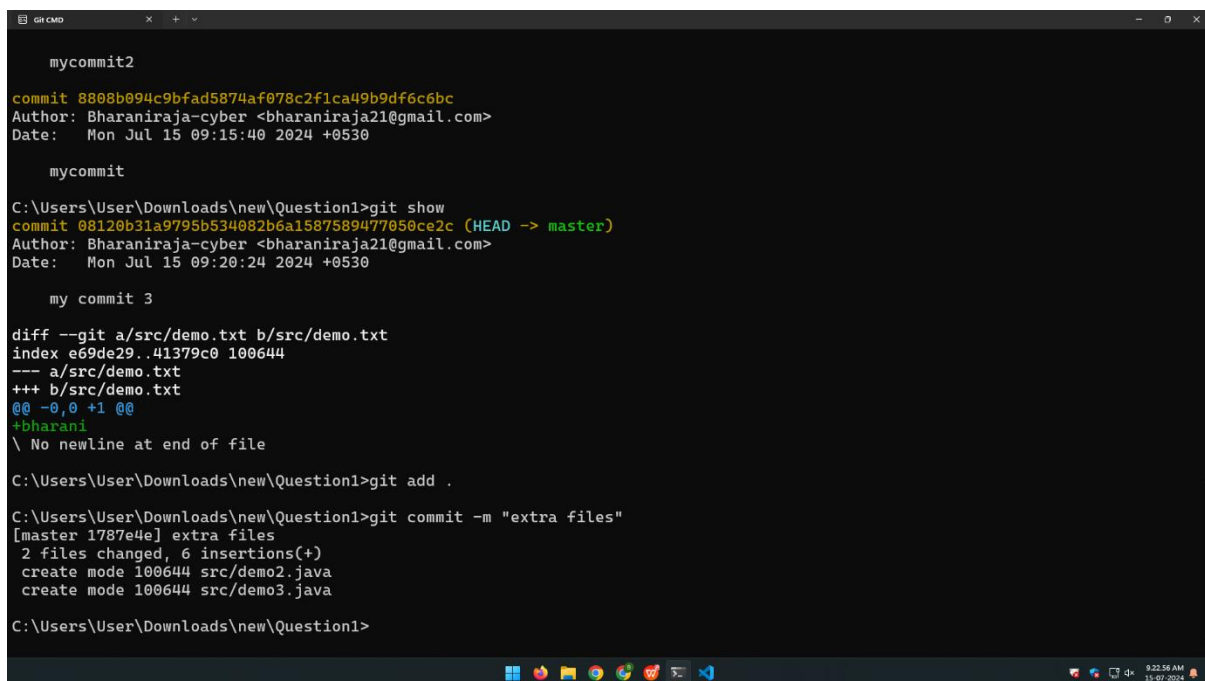
Use 8-10 characters for large repositories:

For example:

`Git show 2f7a467d02`

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

I created two java files and move it to staging area and commit them.



```
mycommit2
commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530

mycommit

C:\Users\User\Downloads\new\Question1>git show
commit 08120b31a9795b534082b6a1587589477050ce2c (HEAD -> master)
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:20:24 2024 +0530

my commit 3

diff --git a/src/demo.txt b/src/demo.txt
index e69de29..41379c0 100644
--- a/src/demo.txt
+++ b/src/demo.txt
@@ -0,0 +1 @@
+bharani
\ No newline at end of file

C:\Users\User\Downloads\new\Question1>git add .

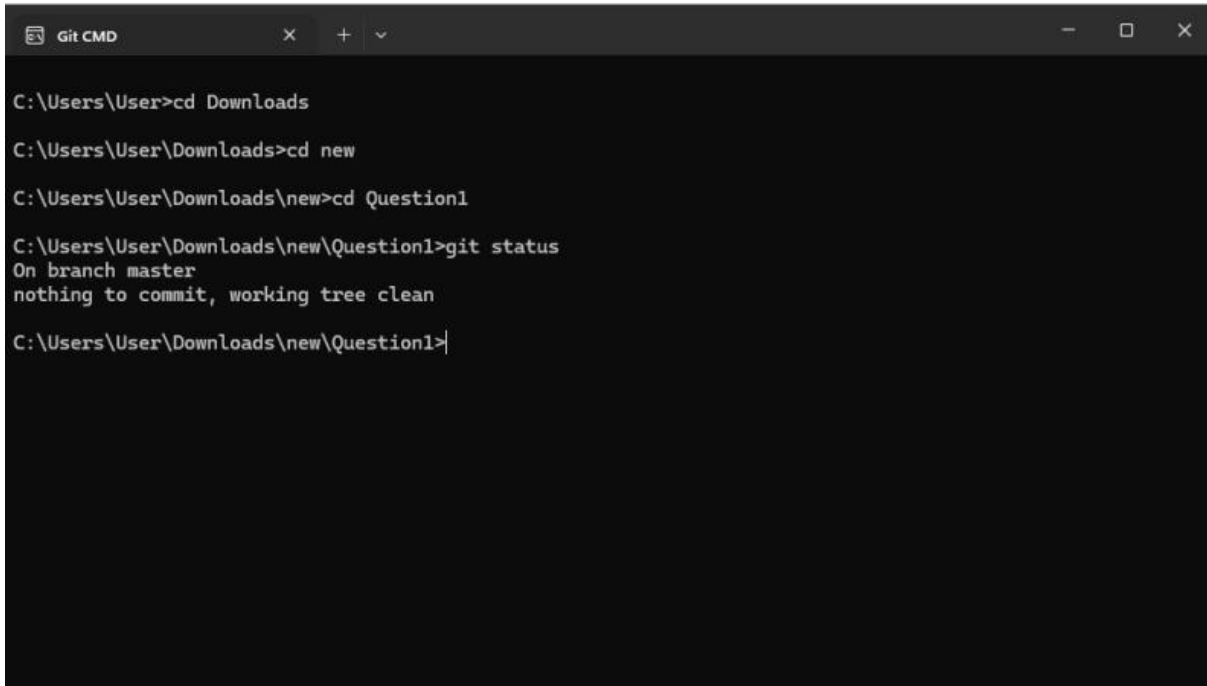
C:\Users\User\Downloads\new\Question1>git commit -m "extra files"
[master 178704e] extra files
2 files changed, 6 insertions(+)
create mode 100644 src/demo2.java
create mode 100644 src/demo3.java

C:\Users\User\Downloads\new\Question1>
```



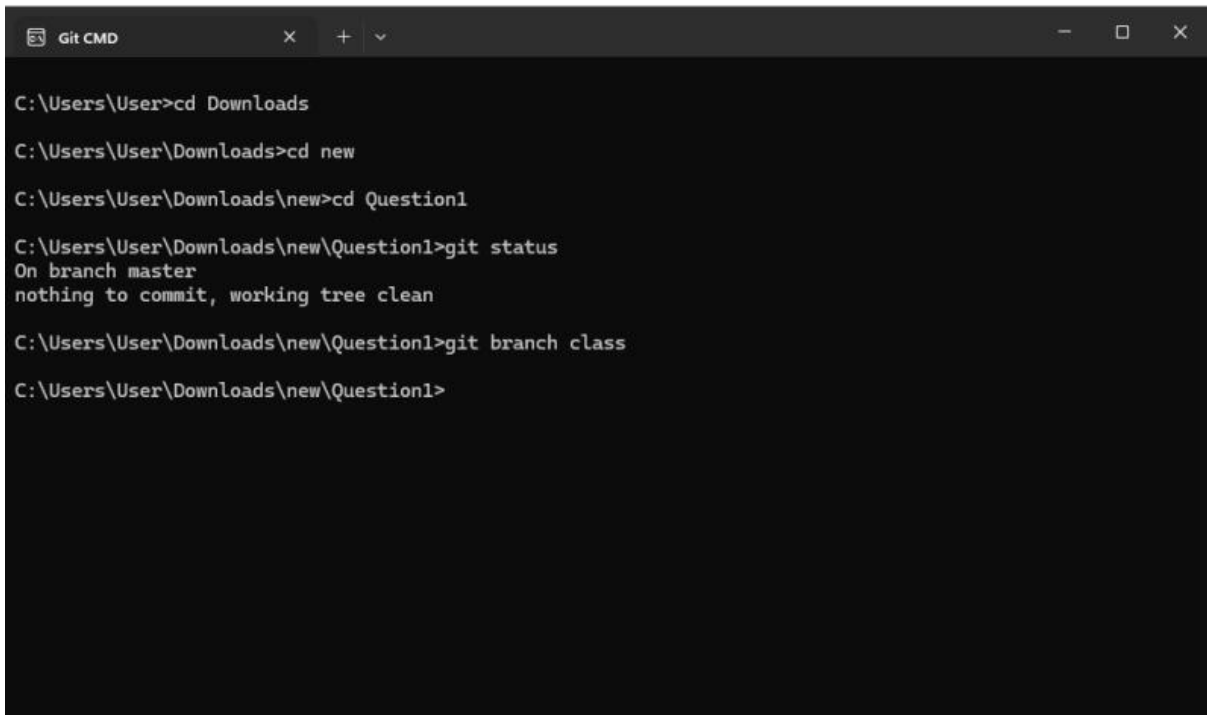
## Exercise-2:

1. Run the **status** command. Notice how it tells you what branch you are in.



```
Git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git status
On branch master
nothing to commit, working tree clean
C:\Users\User\Downloads\new\Question1>
```

2. Use the **branch** command to create a new branch.



```
Git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git status
On branch master
nothing to commit, working tree clean
C:\Users\User\Downloads\new\Question1>git branch class
C:\Users\User\Downloads\new\Question1>
```

3. Use the **checkout** command to switch to it.

```
git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git status
On branch master
nothing to commit, working tree clean
C:\Users\User\Downloads\new\Question1>git branch class
C:\Users\User\Downloads\new\Question1>git checkout class
Switched to branch 'class'
C:\Users\User\Downloads\new\Question1>
```

4. Make a couple of commits in the branch – perhaps adding a new file and/or editing existing ones.

```
git CMD
C:\Users\User>cd Downloads
C:\Users\User\Downloads>cd new
C:\Users\User\Downloads\new>cd Question1
C:\Users\User\Downloads\new\Question1>git status
On branch master
nothing to commit, working tree clean
C:\Users\User\Downloads\new\Question1>git branch class
C:\Users\User\Downloads\new\Question1>git checkout class
Switched to branch 'class'
C:\Users\User\Downloads\new\Question1>git add .
C:\Users\User\Downloads\new\Question1>git commit -m "my commit 4"
[class 1f45943] my commit 4
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/file1.c
create mode 100644 src/sat.txt
C:\Users\User\Downloads\new\Question1>
```

5. Use the **log** command to see the latest commits. The two you just made should be at the top of the list.

```
git CMD
C:\Users\User\Downloads\new\Question1>git add .
C:\Users\User\Downloads\new\Question1>git commit -m "my commit 4"
[class 1f45943] my commit 4
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/file1.c
create mode 100644 src/sat.txt

C:\Users\User\Downloads\new\Question1>git log
commit 1f459431c3d80cb5cb827975293d88ddd8646be2 (HEAD -> class)
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:29:32 2024 +0530

    my commit 4

commit 1787e4eb7183f35bc787eeec5f74576effc13df7 (master)
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:22:54 2024 +0530

    extra files

commit 08120b31a9795b534082b6a1587589477050ce2c
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:20:24 2024 +0530

    my commit 3

commit 946cc271da8214878aee4b34d8bc5efefac1d22d
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:18:11 2024 +0530

    mycommit2

commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530
```

6. Use the **checkout** command to switch back to the master branch. Run **log** again. Notice your commits don't show up now. Check the files also – they should have their original contents.

```
mycommit2

commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530

mycommit

C:\Users\User\Downloads\new\Question1>git checkout master
Switched to branch 'master'

C:\Users\User\Downloads\new\Question1>git log
commit 1787e4eb7183f35bc787eeec5f74576effc13df7 (HEAD -> master)
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:22:54 2024 +0530

    extra files

commit 08120b31a9795b534082b6a1587589477050ce2c
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:20:24 2024 +0530

    my commit 3

commit 946cc271da8214878aee4b34d8bc5efefac1d22d
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:18:11 2024 +0530

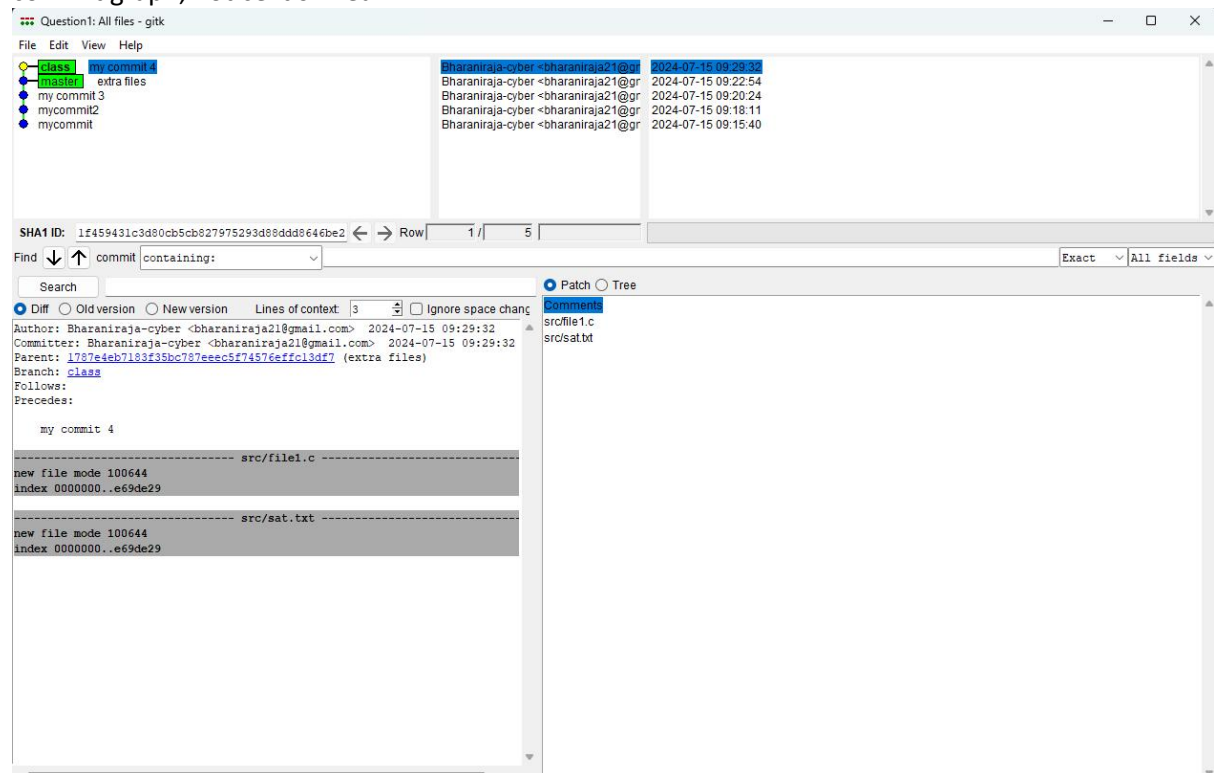
    mycommit2

commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530

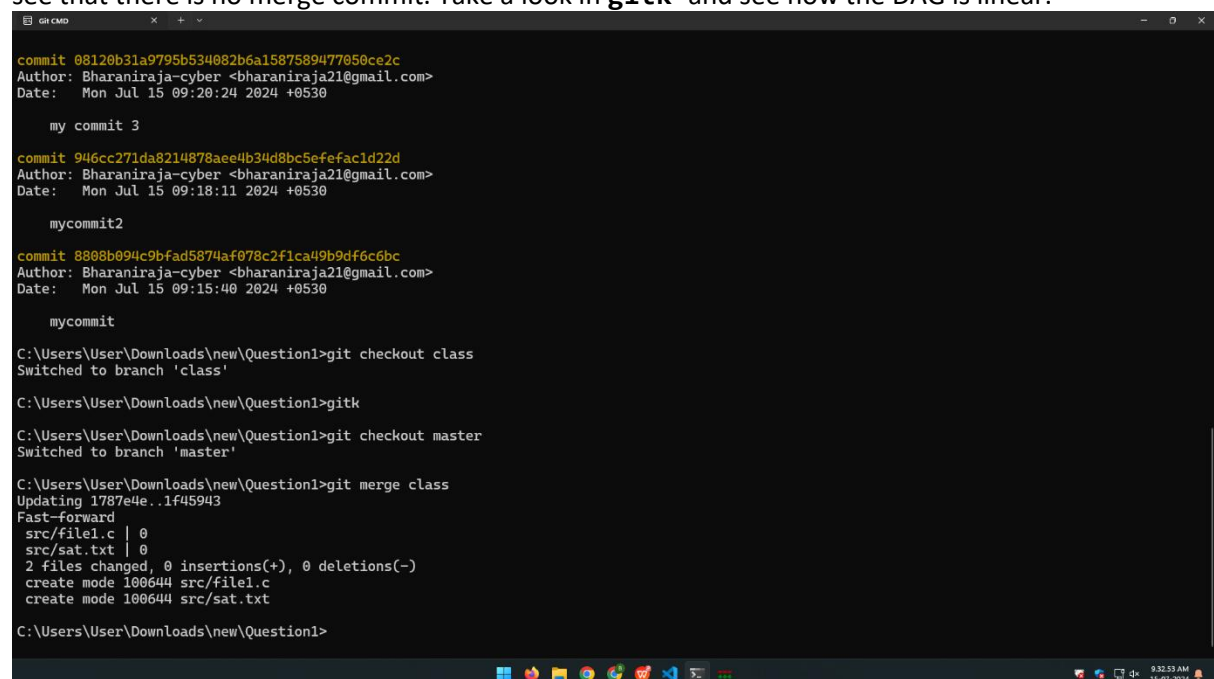
mycommit

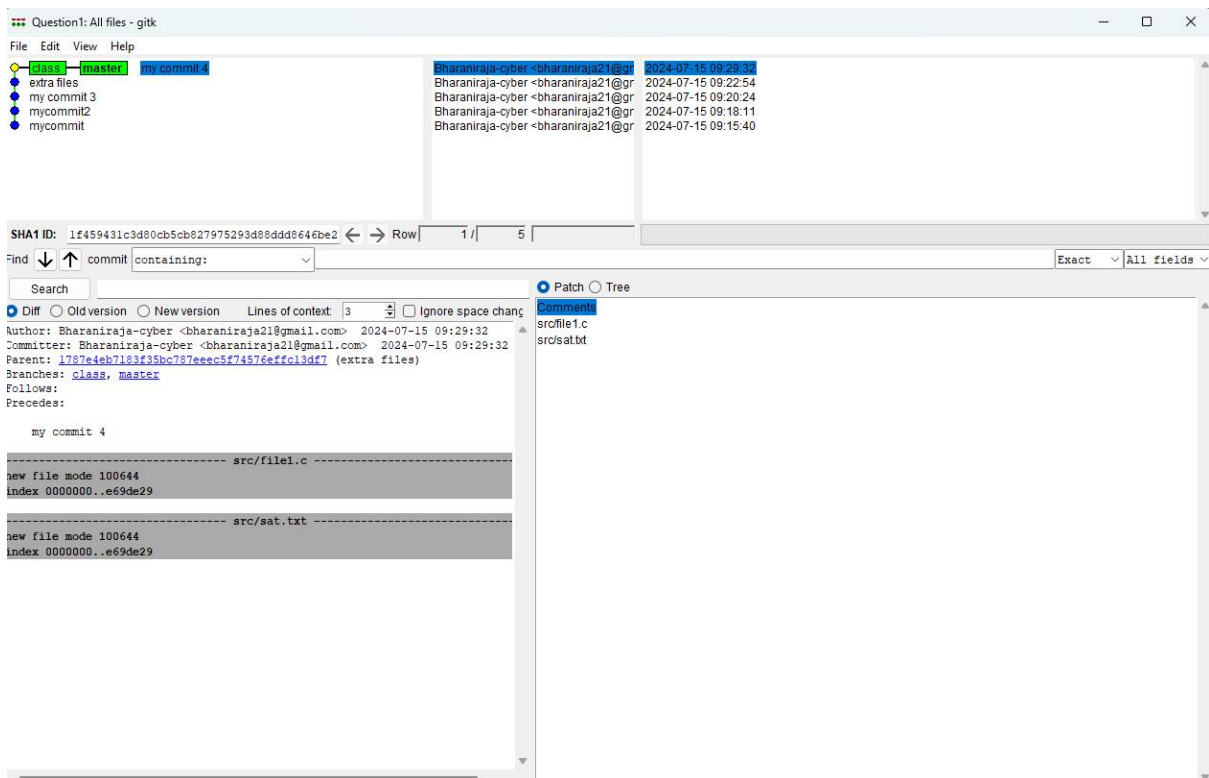
C:\Users\User\Downloads\new\Question1>
```

7. Use the **checkout** command to switch back to your branch. Use **gitk** to take a look at the commit graph; notice it's linear.

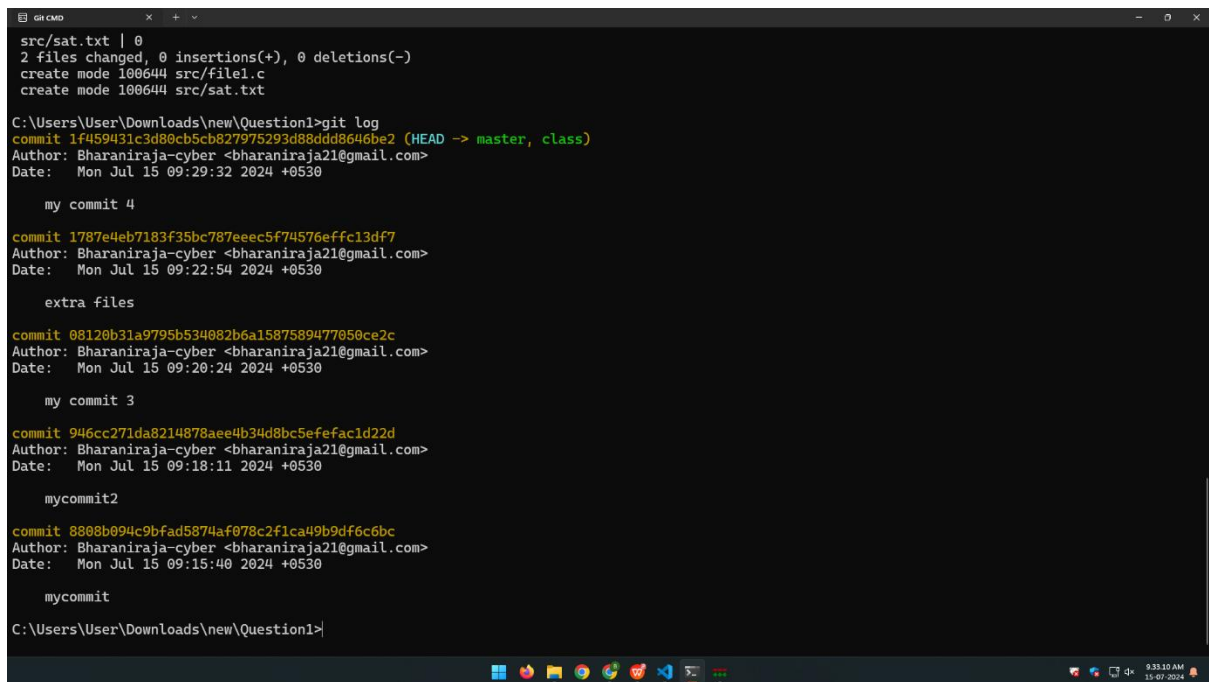


8. Now **checkout** the master branch again. Use the **merge** command to merge your branch in to it. Look for information about it having been a fast-forward merge. Look at **git log**, and see that there is no merge commit. Take a look in **gitk** and see how the DAG is linear.





9. Switch back to your branch. Make a couple more commits.



10. Switch back to master. Make a commit there, which should edit a different file from the ones you touched in your branch – to be sure there is no conflict.

```
git commit
commit 08120b31a9795b534082b6a1587589477050ce2c
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:20:24 2024 +0530

    my commit 3

commit 946cc271da8214878aee4b34d8bc5efefac1d22d
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:18:11 2024 +0530

    mycommit2

commit 8808b094c9bfad5874af078c2f1ca49b9df6c6bc
Author: Bharaniraja-cyber <bharaniraja21@gmail.com>
Date: Mon Jul 15 09:15:40 2024 +0530

    mycommit

C:\Users\User\Downloads\new\Question1>gitk

C:\Users\User\Downloads\new\Question1>git checkout class
Switched to branch 'class'

C:\Users\User\Downloads\new\Question1>git add .

C:\Users\User\Downloads\new\Question1>git commit -m "my commit 5"
[class 17b7922] my commit 5
2 files changed, 2 insertions(+)

C:\Users\User\Downloads\new\Question1>git add .

C:\Users\User\Downloads\new\Question1>git commit -m "alter the sat.txt"
[class e82da02] alter the sat.txt
1 file changed, 1 deletion(-)

C:\Users\User\Downloads\new\Question1>
```

11. Now **merge** your branch again. (Aside: you don't need to do anything to inform Git that you only want to merge things added since your previous merge. Due to the way Git works, that kind of issue simply does not come up, unlike in early versions of Subversion.)

```
git commit
[master 2f7a467] the sat.txt
1 file changed, 1 insertion(+)

C:\Users\User\Downloads\new\Question1>git merge class
Merge made by the 'ort' strategy.
 src/file1.c | 1 +
 1 file changed, 1 insertion(+)

C:\Users\User\Downloads\new\Question1>git add .

C:\Users\User\Downloads\new\Question1>git commit -m "alter the sat.txt"
[master 9eb38a1] alter the sat.txt
1 file changed, 1 deletion(-)

C:\Users\User\Downloads\new\Question1>git merge class
Already up to date.

C:\Users\User\Downloads\new\Question1>git checkout class
Switched to branch 'class'

C:\Users\User\Downloads\new\Question1>git add .

C:\Users\User\Downloads\new\Question1>git commit -m "my commit 6"
[class 6b4a6bc] my commit 6
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/sat2.txt

C:\Users\User\Downloads\new\Question1>git checkout master
Switched to branch 'master'

C:\Users\User\Downloads\new\Question1>git merge class
Merge made by the 'ort' strategy.
 src/sat2.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/sat2.txt

C:\Users\User\Downloads\new\Question1>
```

12. Look at **git log**. Notice that there is a merge commit. Also look in **gitk**. Notice the DAG now shows how things forked, and then were joined up again by a merge commit.

The screenshot shows the gitk GUI with the following components:

- Commit History Graph:** A graph showing the relationship between commits. The 'master' branch is highlighted in green, and the 'class' branch is highlighted in blue. The merge commit 'my commit 6' is shown as a blue dot with two parents: 'my commit 5' (on master) and 'my commit 4' (on class).
- Commit List:** A table of commits with columns for author, email, and date.

Author	Email	Date
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:38:54
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:38:37
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:37:44
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:37:05
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:35:32
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:35:04
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:35:56
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:29:32
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:22:54
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:20:24
Bharaniraja-cyber	<bharaniraja21@gmail.com>	2024-07-15 09:18:11
- SHA1 ID:** c30178336f5910eb5735772b2f8627d09032aecd
- Find:** A search bar with the text 'commit' and a dropdown menu.
- Diff View:** A view showing the differences between the current commit and its parent. The 'Diff' view is selected, and the 'Patch' view is also visible. The diff shows the merge of the 'class' branch into 'master'.