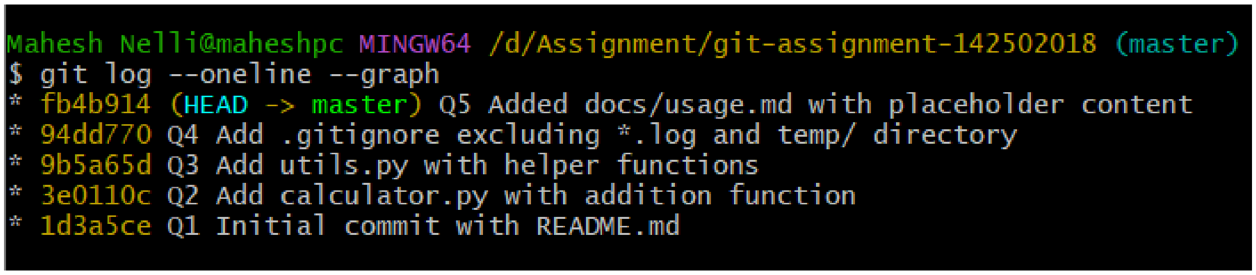
# Mlops Assignment-2

## Question 1:

**Repository Creation:** Create a repository named ***git-assignment-[your-roll-no]*** within the **assignment folder as discussed in the class** with the following specific initial structure and required Initial Commits (must be created in this exact order):
a. Initial commit with **README.md** containing your roll number and current timestamp
b. Add **src/calculator.py** with basic addition function
c. Add **src/utils.py** with helper functions
d. Create **.gitignore** excluding ***.log and temp/ directory**
e. Add **docs/usage.md** with placeholder content
Do the following:
a. Take the screenshot of the command git log --oneline --graph showing exact commit sequence
b. Each commit message must follow the format: [***Qno] Description***. Here Qno for first question is Q1, for second it is Q2 and so on.

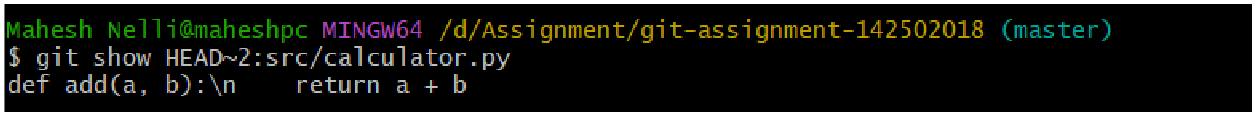
**Output Screenshot:**



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --oneline --graph
* fb4b914 (HEAD -> master) Q5 Added docs/usage.md with placeholder content
* 94dd770 Q4 Add .gitignore excluding *.log and temp/ directory
* 9b5a65d Q3 Add utils.py with helper functions
* 3e0110c Q2 Add calculator.py with addition function
* 1d3a5ce Q1 Initial commit with README.md
```

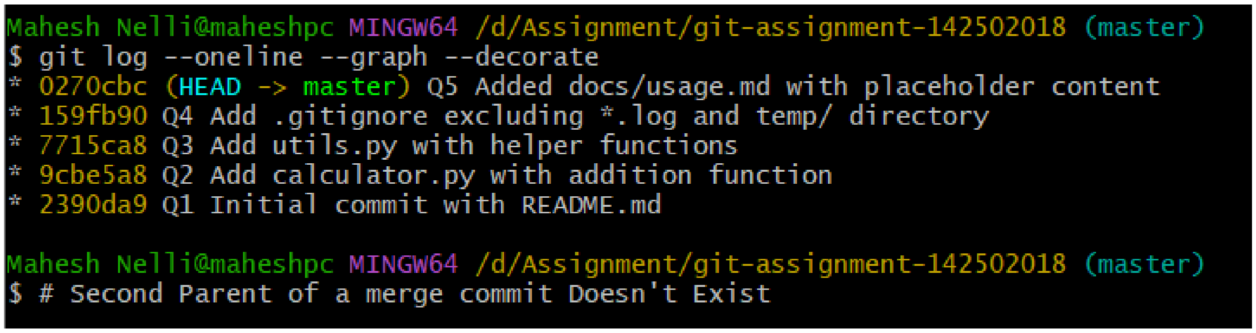
## Question 2:

2.a) Show the content of calculator.py from 2 commits before HEAD



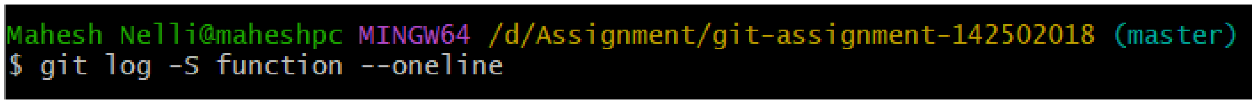
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git show HEAD~2:src/calculator.py
def add(a, b):\n    return a + b
```

2.b) Display the commit message of the second parent of a merge commit (if exists)



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --oneline --graph --decorate
* 0270cbc (HEAD -> master) Q5 Added docs/usage.md with placeholder content
* 159fb90 Q4 Add .gitignore excluding *.log and temp/ directory
* 7715ca8 Q3 Add utils.py with helper functions
* 9cbe5a8 Q2 Add calculator.py with addition function
* 2390da9 Q1 Initial commit with README.md

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ # Second Parent of a merge commit Doesn't Exist
```

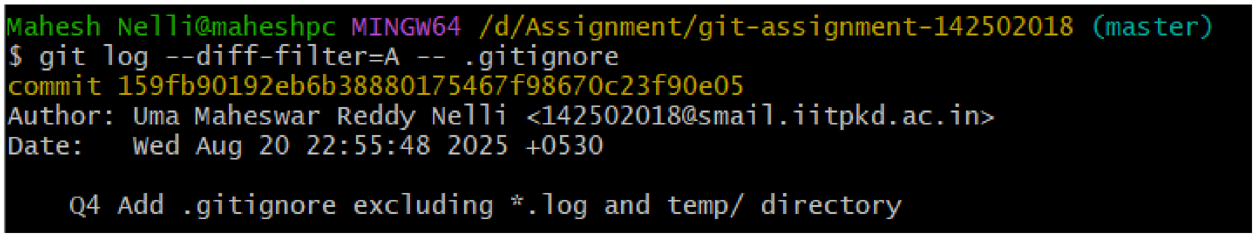
2.c) Find the commit that introduced the word "**function**" in any file



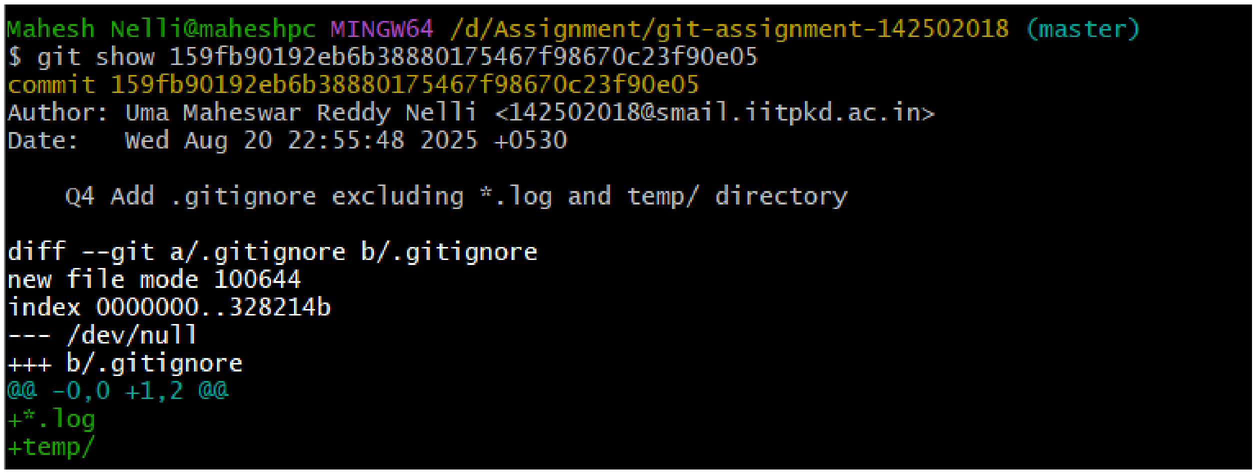
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log -S function --oneline
```

2.d) Show differences between the commit where .gitignore was added and its

immediate predecessor

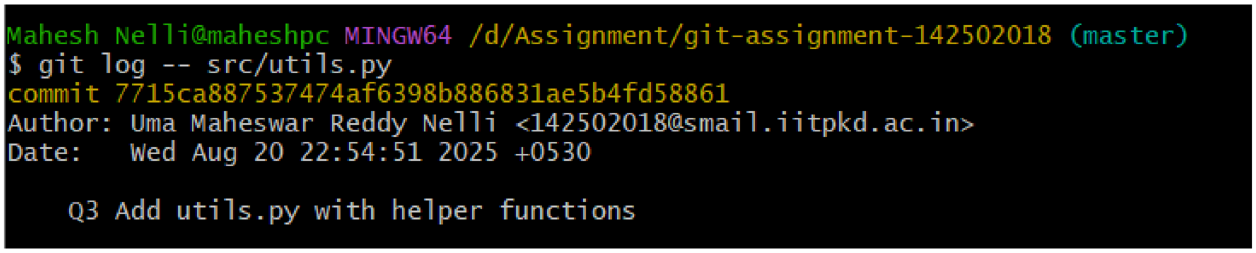


```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --diff-filter=A -- .gitignore
commit 159fb90192eb6b38880175467f98670c23f90e05
Author: Uma Maheswar Reddy Nelli <142502018@smail.iitpkd.ac.in>
Date:   Wed Aug 20 22:55:48 2025 +0530

    Q4 Add .gitignore excluding *.log and temp/ directory
```



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git show 159fb90192eb6b38880175467f98670c23f90e05
commit 159fb90192eb6b38880175467f98670c23f90e05
Author: Uma Maheswar Reddy Nelli <142502018@smail.iitpkd.ac.in>
Date:   Wed Aug 20 22:55:48 2025 +0530

    Q4 Add .gitignore excluding *.log and temp/ directory

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..328214b
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,2 @@
+*.log
+temp/
```

2.e) List all commits that modified **src/utils.py** specifically

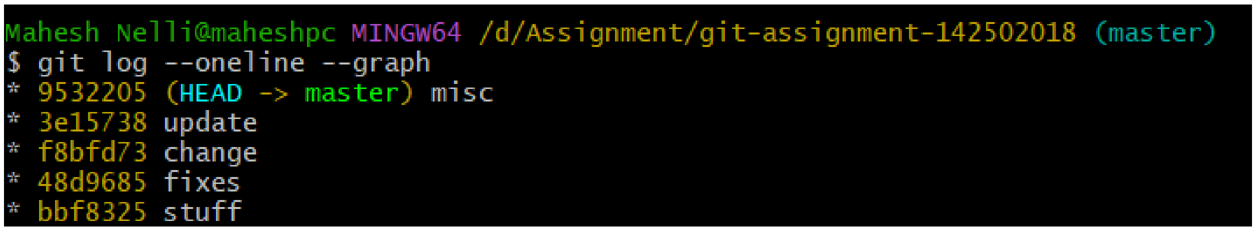


```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log -- src/utils.py
commit 7715ca887537474af6398b886831ae5b4fd58861
Author: Uma Maheswar Reddy Nelli <142502018@smail.iitpkd.ac.in>
Date:   Wed Aug 20 22:54:51 2025 +0530

    Q3 Add utils.py with helper functions
```

## Question 3:

3.a) Create 5 commits with intentionally poor commit messages like "stuff",

"fixes", "change



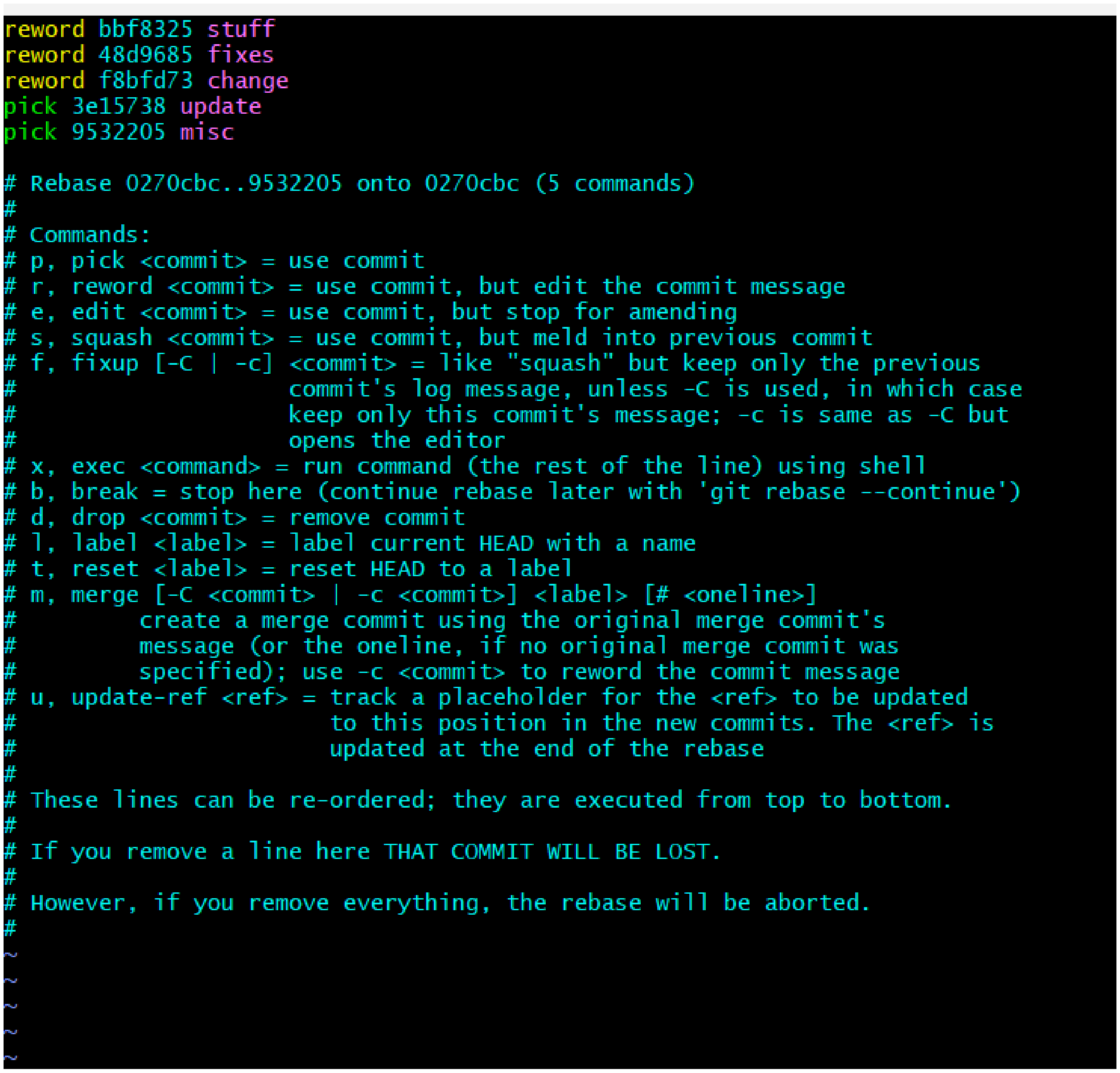
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --oneline --graph
* 9532205 (HEAD -> master) misc
* 3e15738 update
* f8bfd73 change
* 48d9685 fixes
* bbf8325 stuff
```

3.b) Use interactive rebase to:

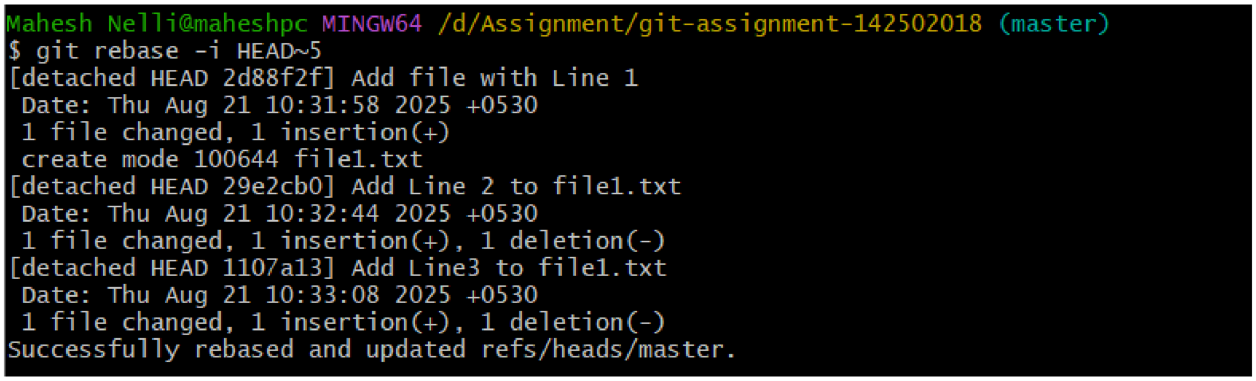
i. Rewrite 3 commit messages to be descriptive

ii. Squash 2 commits into 1

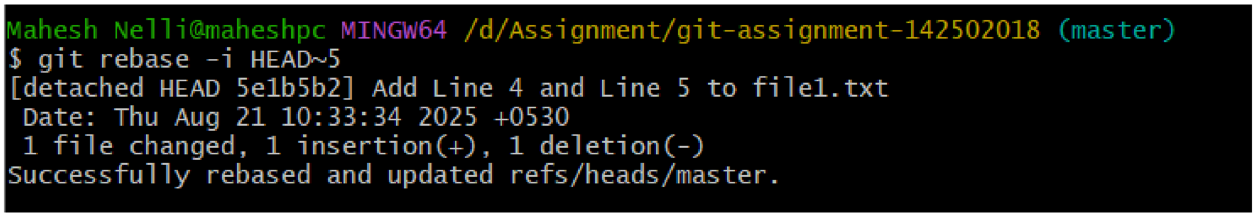
iii. Reorder commits to make logical sense



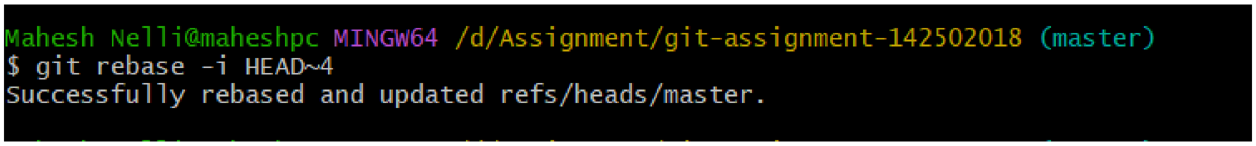
```
reword bbf8325 stuff
reword 48d9685 fixes
reword f8bfd73 change
pick 3e15738 update
pick 9532205 misc

# Rebase 0270cbc..9532205 onto 0270cbc (5 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
#         create a merge commit using the original merge commit's
#         message (or the oneline, if no original merge commit was
#         specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                       to this position in the new commits. The <ref> is
#                       updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
~
~
~
~
~
```



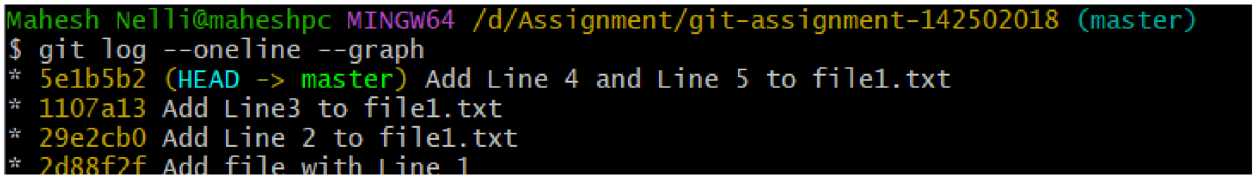
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git rebase -i HEAD~5
[detached HEAD 2d88f2f] Add file with Line 1
 Date: Thu Aug 21 10:31:58 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
[detached HEAD 29e2cb0] Add Line 2 to file1.txt
 Date: Thu Aug 21 10:32:44 2025 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
[detached HEAD 1107a13] Add Line3 to file1.txt
 Date: Thu Aug 21 10:33:08 2025 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/master.
```



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git rebase -i HEAD~5
[detached HEAD 5e1b5b2] Add Line 4 and Line 5 to file1.txt
 Date: Thu Aug 21 10:33:34 2025 +0530
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/master.
```



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git rebase -i HEAD~4
Successfully rebased and updated refs/heads/master.
```



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --oneline --graph
* 5e1b5b2 (HEAD -> master) Add Line 4 and Line 5 to file1.txt
* 1107a13 Add Line3 to file1.txt
* 29e2cb0 Add Line 2 to file1.txt
* 2d88f2f Add file with Line 1
```

3.c) Document each step of the interactive rebase process

1. **Initial History** – I had 5 commits with poor messages like *“stuff”*, *“fixes”*, *“change”, ”update” and “misc”.*
2. **Started Interactive Rebase** – Used git rebase -i HEAD~5 to open the last 5 commits.
3. **Edited Rebase Plan** –

   Changed 3 commits from pick to reword to rewrite messages.

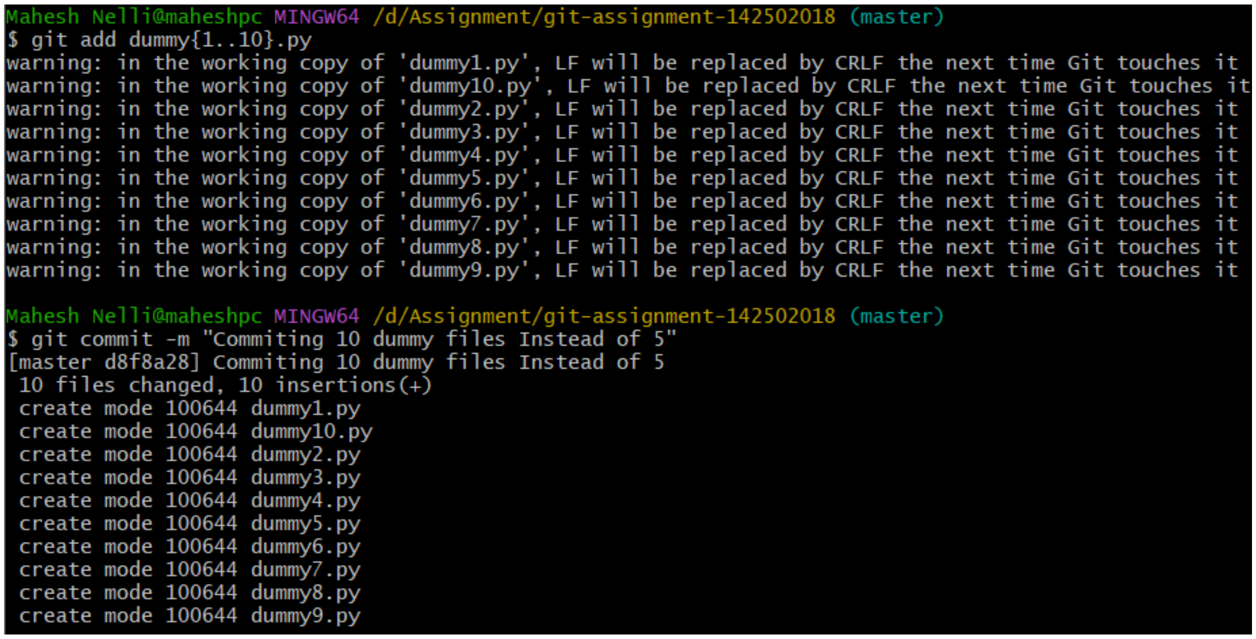
   Changed 1 commit from pick to squash to merge it into the commit above.

   Reordered lines to make commits follow logical order.
4. **Updated Commit Messages** – Git prompted me to write new descriptive commit messages for the reword and to merge messages for the squash.

## **Question 4**:

4.a. Accidentally commit 10 files where only 5 should be committed (You can

use dummy files as well such as dummy1.py, dummy2.py, etc



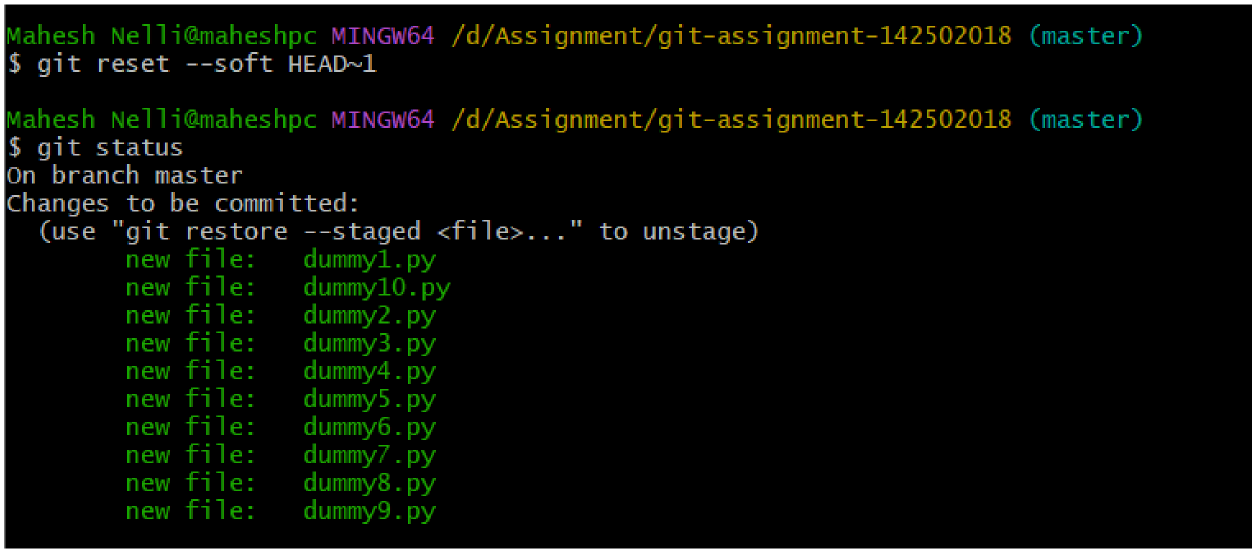
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git add dummy{1..10}.py
warning: in the working copy of 'dummy1.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy10.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy2.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy3.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy4.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy5.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy6.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy7.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy8.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy9.py', LF will be replaced by CRLF the next time Git touches it

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git commit -m "Commiting 10 dummy files Instead of 5"
[master d8f8a28] Commiting 10 dummy files Instead of 5
 10 files changed, 10 insertions(+)
 create mode 100644 dummy1.py
 create mode 100644 dummy10.py
 create mode 100644 dummy2.py
 create mode 100644 dummy3.py
 create mode 100644 dummy4.py
 create mode 100644 dummy5.py
 create mode 100644 dummy6.py
 create mode 100644 dummy7.py
 create mode 100644 dummy8.py
 create mode 100644 dummy9.py
```

4.b) The commit was NOT pushed yet

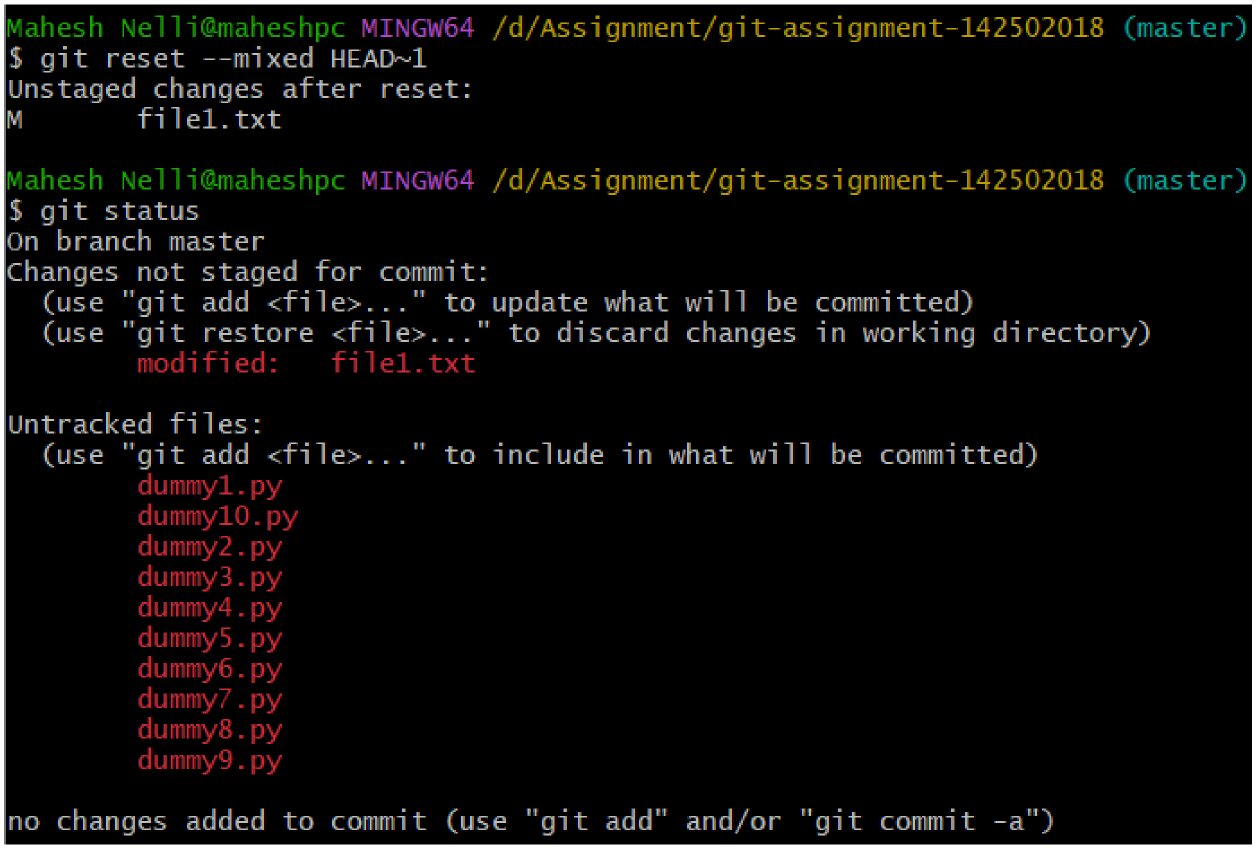
4.c) Use git reset to uncommit, then re-commit correctly

4.d) Show working directory and staging area status throughout

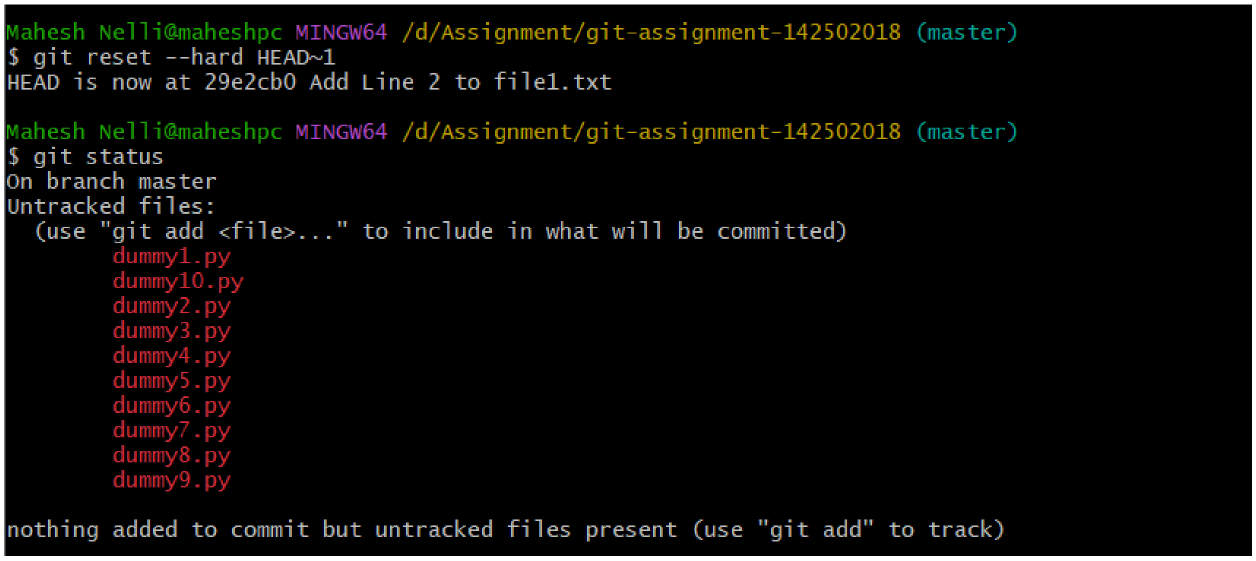
4.e) Show difference between **git reset --soft, --mixed, and --hard**



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git reset --soft HEAD~1

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   dummy1.py
        new file:   dummy10.py
        new file:   dummy2.py
        new file:   dummy3.py
        new file:   dummy4.py
        new file:   dummy5.py
        new file:   dummy6.py
        new file:   dummy7.py
        new file:   dummy8.py
        new file:   dummy9.py
```



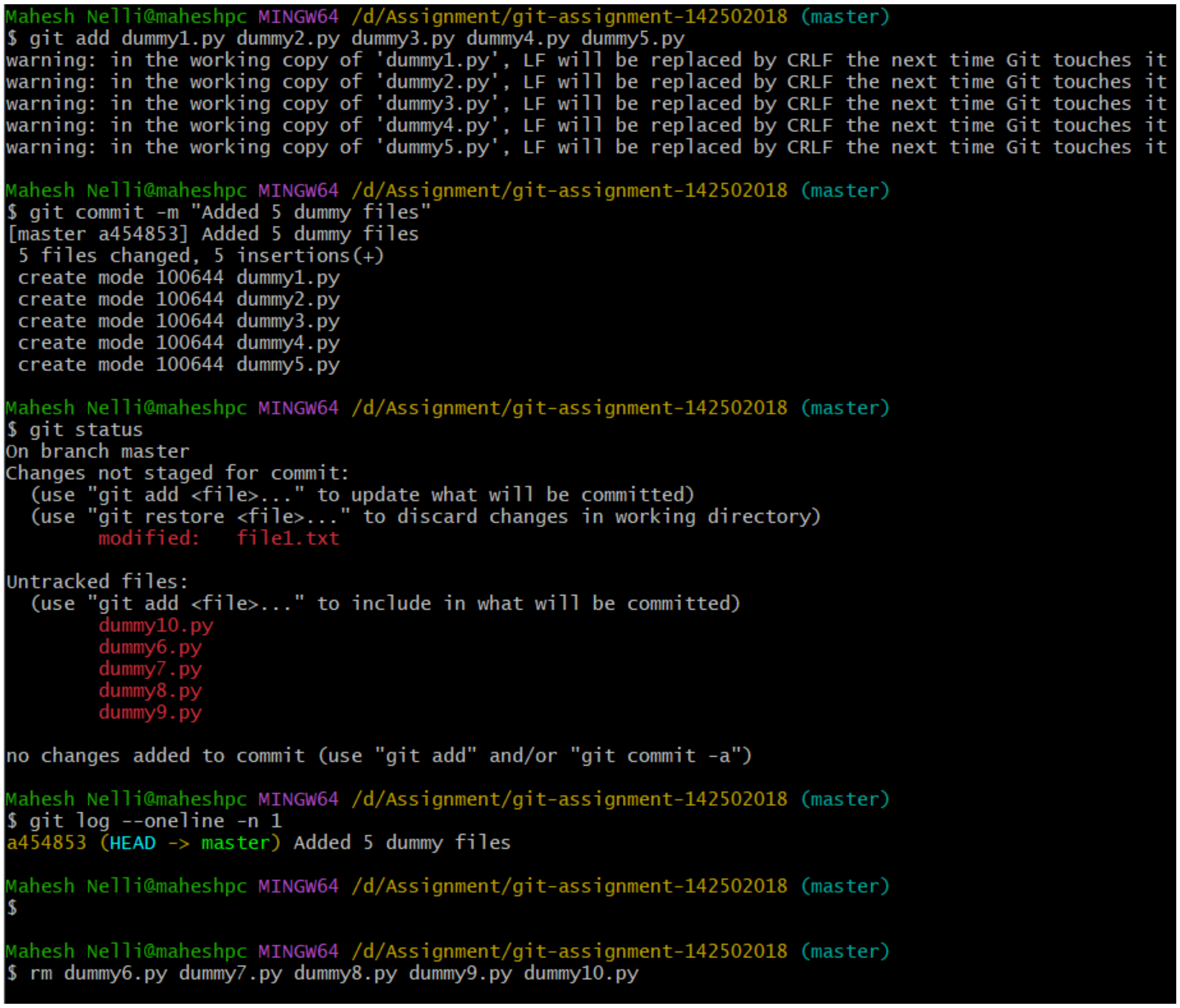
```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git reset --mixed HEAD~1
Unstaged changes after reset:
M       file1.txt

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dummy1.py
        dummy10.py
        dummy2.py
        dummy3.py
        dummy4.py
        dummy5.py
        dummy6.py
        dummy7.py
        dummy8.py
        dummy9.py

no changes added to commit (use "git add" and/or "git commit -a")
```



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git reset --hard HEAD~1
HEAD is now at 29e2cb0 Add Line 2 to file1.txt

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dummy1.py
        dummy10.py
        dummy2.py
        dummy3.py
        dummy4.py
        dummy5.py
        dummy6.py
        dummy7.py
        dummy8.py
        dummy9.py

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

- **Soft reset (git reset --soft <commit>)** : Moves HEAD to the specified commit but keeps all changes staged (in index), ready to be committed again.
- **Mixed reset (git reset --mixed <commit>):** *(default)* → Moves HEAD to the commit, keeps changes in working directory but unstaged, so you need to git add again.
- **Hard reset (git reset --hard <commit>)** : Moves HEAD to the commit and discards all changes in staging and working directory, making everything exactly like that commit.

**Screenshot of Committing correct 5 Files:**



```
Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git add dummy1.py dummy2.py dummy3.py dummy4.py dummy5.py
warning: in the working copy of 'dummy1.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy2.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy3.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy4.py', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'dummy5.py', LF will be replaced by CRLF the next time Git touches it

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git commit -m "Added 5 dummy files"
[master a454853] Added 5 dummy files
 5 files changed, 5 insertions(+)
 create mode 100644 dummy1.py
 create mode 100644 dummy2.py
 create mode 100644 dummy3.py
 create mode 100644 dummy4.py
 create mode 100644 dummy5.py

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dummy10.py
        dummy6.py
        dummy7.py
        dummy8.py
        dummy9.py

no changes added to commit (use "git add" and/or "git commit -a")

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ git log --oneline -n 1
a454853 (HEAD -> master) Added 5 dummy files

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$

Mahesh Nelli@maheshpc MINGW64 /d/Assignment/git-assignment-142502018 (master)
$ rm dummy6.py dummy7.py dummy8.py dummy9.py dummy10.py
```