**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:

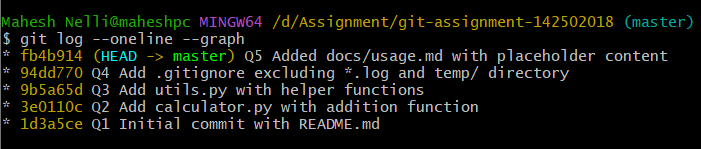
a. Take the screenshot of the command git log --oneline --graph showing exact

commit sequence

b. Each commit message must follow the format: [***Qno] Description***. Here Qno for

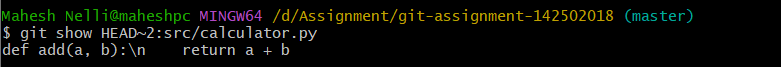
first question is Q1, for second it is Q2 and so on.

**Output Screenshot:**

****

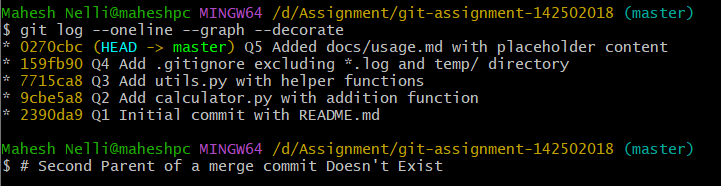
**Question 2:**

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



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

exists)

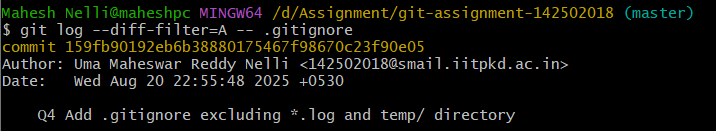


2.c) Find the commit that introduced the word "**function**" in any file



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

immediate predecessor



A computer screen with white and yellow text

AI-generated content may be incorrect.

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

A screenshot of a computer program

AI-generated content may be incorrect.

**Question 3:**

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

"fixes", "change

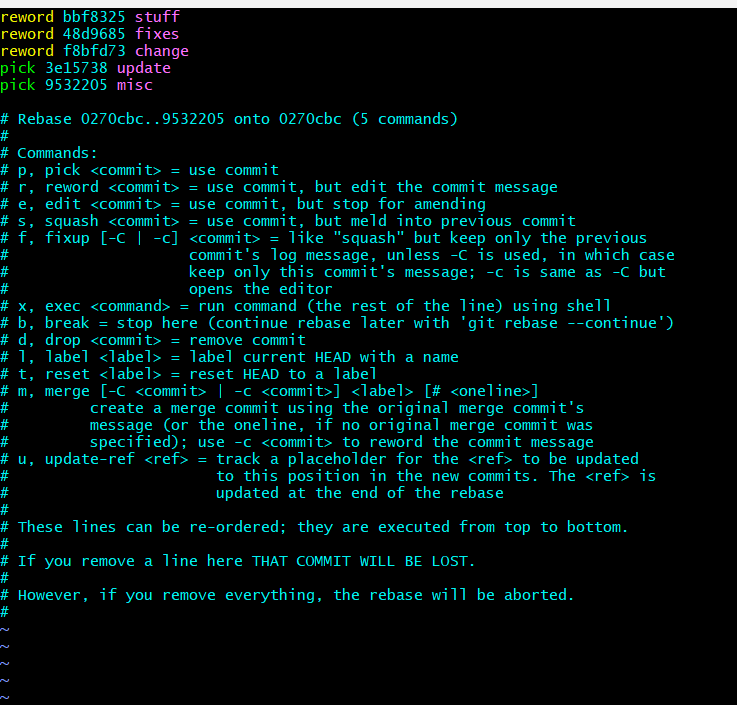


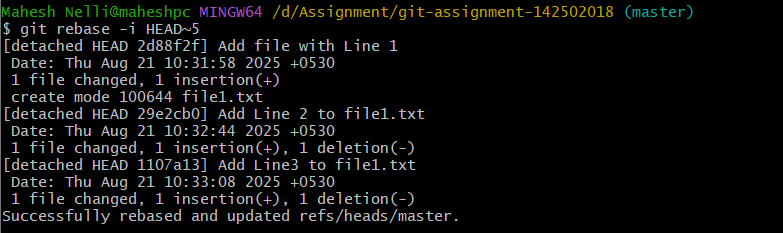
3.b) Use interactive rebase to:

i. Rewrite 3 commit messages to be descriptive

ii. Squash 2 commits into 1

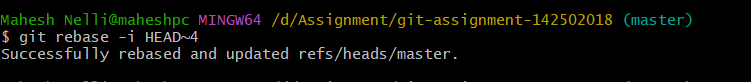
iii. Reorder commits to make logical sense





A screen shot of a computer

AI-generated content may be incorrect.



A black screen with yellow and white text

AI-generated content may be incorrect.

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.

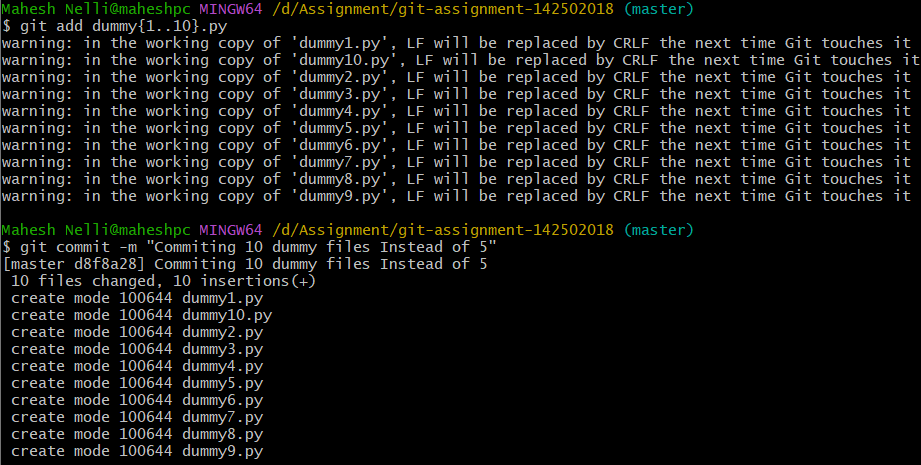
Reordered lines to make commits follow logical order.

1. **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

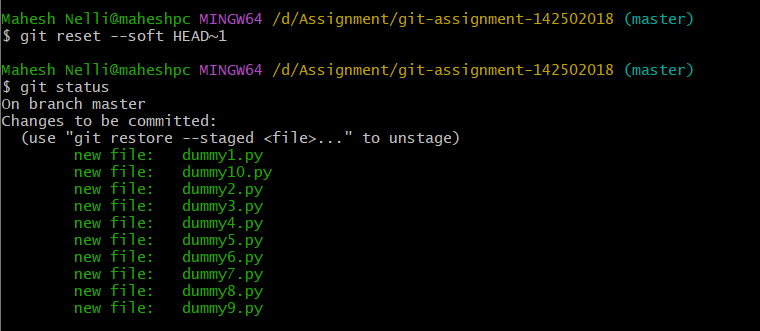


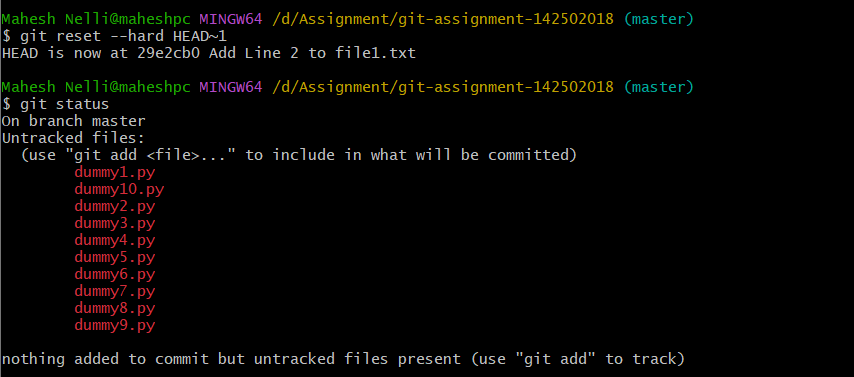
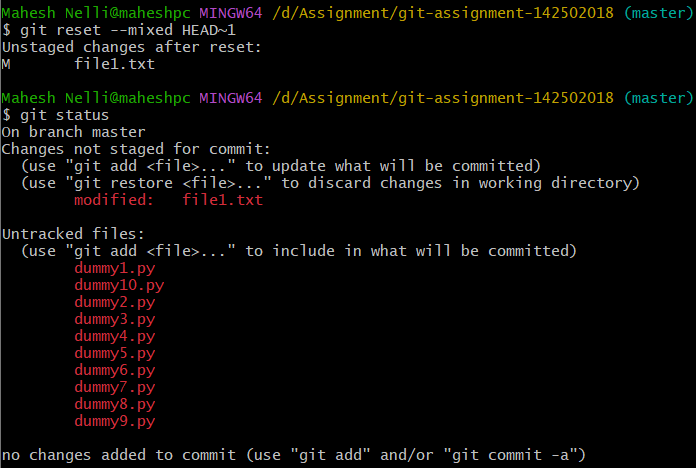
4.b) The commit was NOT pushed yet

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**





* **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:**

