3/9/24 Program 1 :

## Setting up & Basic Commands :

1. To initialize a new Git repository in a directory, create a new file, add it to the staging area, & commit the changes with an appropriate commit message, you can follow these steps :

1. Initialize a new Git repository :

```
git init
```

2. Create a new file :

```
touch your_file_name.txt
```

pr1.txt

3. Add the file to the staging area :

```
git add your_file_name.txt
```

pr1.txt

4. Commit the changes with a message :

```
git commit -m "Initial commit : Add your_file_name.txt"
```

pr1.txt

Output :-

```
PC0314@DESKTOP-SJLQKHQ MINGW64
  ~/gitlab (master)
$ git init
Initialized empty Git repository in
  C:/Users/PC0314/gitlab/.git/

PC0314@DESKTOP-SJLQKHQ MINGW64
  ~/gitlab (master)
$ touch pr1.txt

PC0314@DESKTOP-SJLQKHQ MINGW64
  ~/gitlab (master)
$ git add pr1.txt

PC0314@DESKTOP-SJLQKHQ MINGW64
  ~/gitlab (master)
$ git commit -m "initial commit message"
[master (root-commit) 8b188e8] initial
commit message
1 file changed, 0 insertions (+),
0 deletions (-)
create mode 100644 pr1.txt
```

10/9/24 Program : 2

## Creating and Managing Branches

2 Create a new branch name 'feature-branch' Switch to the 'master' branch. Merge the 'feature-branch' into 'master'

1 Create a new branch :

```
git branch feature-branch
```

2 Switch to the master branch :

```
git checkout master
```

Alternatively, you can use more recent 'git switch' command :

```
git switch master
```

3 Merge the feature-branch into master :

```
git Merge feature-branch
```

Output :-

```
DELL @ Dell me ~ (master)
$ git branch feature - branch

DELL @ Dell me ~ (master)
$ git checkout feature - branch
Switched to branch 'feature - branch'

DELL @ Dell me ~ ( feature - branch)
$ git checkout master
Switched to branch 'master'

DELL @ Dell me ~ (master)
$ git merge feature - branch
 updating efa847d.. fc46a5a
Fast - forward
Start. txt 10
1 file changed, 0 insertions (+),
   0 deletions (-)
Create mode 100644 Star. txt
```

Elegant Writing

# Program-3

## Creating and Managing branches

03. Write the commands to stash your changes switch branches and then apply the Stashed changes

1. Stash changes :

```
git stash
```

2. Change to the desired branch :

```
git checkout <branch-name>
```

3. Apply the stash :

```
git stash apply
```

Output :-

```
DELL @ Dell me    ~(master)
$ git stash
Saved Working directory and index state
  WIP on master : 953cfa9 message

DELL @ Dell me    ~(master)
$ git checkout moon
Switched to branch 'moon'

DELL @ Dell me    ~(moon)
$ git Stash
Saved Working directory and index state
  WIP on moon : 3cbea09 Commits

DELL @ Dell me  ~(moon)
$ git stash list
Stash @ {0} : WIP on moon: 3cbe09. Commits
Stash@ {1}: WIP on master : 953cfa9 message
Stash @ {2} : on moon : red
Stash @ {3} : on moon : files
Stash @ {4} : on master : files.
```

# Program - 04.

## Collaborations and Remote Repositories :

4. Clone a Remote repository to your local machine.

→ Before cloning the repository. We need to do some steps.

* Create an account in github.
* Create a repository in GITHUB with any name XYZ.
* then copy that URL address paste it in git Clone Command.

- Clone the repository :
Use the 'git clone' Command to clone the remote repository. Replace '<repository-url>' with the actual URL of the Git repository you want to clone. you can obtain the URL from the repository's hosting service.

```
git clone <repository-URL>
```

For Example :

```
git clone https://github.com/example/repo.git
```

Output :-

```
DELL @ Dell me ~ (moon)
$ git clone https://github.com/sabcse1/
                    git-lab programs

Cloning into 'git-lab'-programs....
remote : Enumerating objects : 3, done,
remote : Counting objects : 100% (3/3),
                    done
remote : Compressing objects : 100% (2/2),
                    done.
remote : Total 3 (delta 0), reused 0
                (delta 0), pack-reused 0
Receiving objects : 100% (3/3), done.
```

# Program - 05

## Collaborations and Remote Repositories :

5. To fetch the latest changes from a remote repo & rebase your local branch onto the updated remote branch, you can follow these steps using Git :

### Solution - A :

- When you run git fetch, it retrieves commits, files, branches, and tags from remote repository.
- The fetched content is isolated from your local development work. It doesn't affect your current work.

### Follow these Steps :

1. First, Ensure you are in your local repo directory using a terminal or Command prompt.
2. Fetch the latest changes from the remote repository.

## git fetch origin

Replace 'origin' with the name of your remote if its different.

git fetch https://github.com/sabcse/lab1.

Output :-

```
DELL @ Dell me ~(master)
$ git fetch https://github.com/sabcse/lab1

remote : Enumerating objects : 9, done
remote : Counting objects : 100% (9/9), done
remote : Compressing objects : 100% (6/6), done
remote : Total 9 (delta 0), reused 0 (delta 0),
         pack - reused 0

From https://github.com/sabcse/lab1
* branch          HEAD      → FETCH-HEAD
```

## Solution B :-

## Rebase Command.

Git rebase can integrate the changes from one branch to another by overcoming the problems that we might have faced while using the git merge cmd. The changes we will do will be recorded in the form of logs which are useful to go through if any mistakes happen.

To rebase the moon branch onto the master branch, you can follow these steps :

1. Checkout the branch you want to rebase onto :

```
git checkout master
```

2. Start the rebase :

```
git rebase moon
```

This will take the changes from the 'moon' branch to replay them on top of the 'master' branch.

After the rebase, you might need to resolve any conflicts that arise. Git will pause at each commit where there's a conflict, allowing you to resolve it manually. After resolving conflicts, you continue the rebase process by executing:

```
git rebase -- Continue
```

Output :-

DELL @ Dell me . ~(moon)
$ git checkout master
Switched to branch master

DELL @ Dell me ~(master)
$ git rebase moon.
Successfully rebased and updated
refs / heads / master

# Program - 06.

## Collaborations and Remote Repositories:

6. Write a Command to merge "feature-branch" into "master" while providing a Custom Commit message for the merge.

### Solution:

To merge the "feature-branch" into "master" while providing a Custom Commit message you can use the following Command.

```
git merge --no --ff -m "Custom Commit message for the merge "feature-branch"
```

* Explanation:

- 'git merge' :- Initiates the merge process
- '--no-ff' :- Forces Git to create a merge commit even if it could perform fast-forward merge.
- "-m" :- Custom Commit message for the

merge specifies the custom commit message for the merge.

- 'feature-branch' :- Specifies the branch you want to merge into the current branch in this case 'master'

Output :-

```
DELL@ Dell me MINGW64 ~(master)
$ git merge --no-ff -m "Custom commit message for the merge 'feature-branch'
Merge made by the 'ort' Strategy
program.txt/0

1 file changed, 0 insertions (+), 0 deletion(-)
create mode 10064 program.txt
```

## Program - 07

### Git Tags and Release

7. Write the command to create a light weight GIT tag named "V1.0" for a commit in your local repository.

```
git tag V1.0
```



Output :-

```
DELL@ Dell me ~ (Master)
$ git tag V1.0

DELL@ Dell me ~ (Master)
$ git tag list

DELL@ Dell me ~ (Master)
$ git show V1.0
Commit e90fec5d840a8935897C303fb51fcadff
(HEAD -> Master , tag : V1.0, tag : list
```

# Program - 08:

## Advanced Git operations

8. Write a command to cherry-pick a range of commits from the source branch to current branch.

```
git cherry-pick <commit-hash>
```

Replace '<commit-hash>' with the hash of the commit you want to cherry-pick.



Output :-

```
DELL @ Dell me ~ (master)
$ git cherry-pick b8f4c4c81086038a74125ccfd1...
[master db336ff] Commiting files
Date : Mon feb 19 13:09:16 2024 +0530
1 file changed, 0 insertions (+), 0 deletions(-)
create mode 100644 file 5.txt
```

# Program - 09

## Analysis and Changing git history.

9. Given a Commit ID, how would you use git to view the details of that specific commit. Including the author, date, and commit message.

```
git Show <Commit - ID>
```

Replace '<Commit - ID>' with the ID of commit you want to inspect
For Example

```
git Show abc123
```



Output :-

```
DELL @ Dell me ~(master)
$ git Show 820a45bcod015f865c2395acodfa
Commit 820a45bcod015f865c2395acodf9
Author : Sabcse < Sabcse 2023 @ gmail.com>
Date : Mon Feb 19 12:23:05 2024 f0530
    Commiting program 2.
diff --git a/ program 2. txt b/program 2.tat
new file mode 100644.
inden 0000000 -- e69de29.
```

Program :- 10.

## Analysis & Changing git history

10. Write the Command to list all Commits made by the author "John Doe" between "2023-1-1" & "2023-12-31"

```
git log --author="JohnDoe"--since="2023-01-01"
  --until "2023-12-31"
```

This Command uses the 'git log' command with the "--author', '--since' and '--until' options to filter the Commits by specified author & within the specified date range.



## Output :

```
DELL @ Dell me ~ (master)
$ git log --author="Sab" --since="2023-1-27"
  --until="2024-2-15"
Commit 9ab421206580 7a392c06fd3doecc8bc48a56
Author : Sab <Sab 30703 @ gmail.com>
Date : Wed feb 14 13:29:37 2024 f0530
      Merging branches
Commit f4cdc614bb8ad3128e268d7d45961ed81c4a
(main, feature-branch, feature)
Author : Sab <Sab 30703 @ gmail.com>
Date : Sun Jan 21 14:16:17 2024 f0530
      Commits
```

## Program - 11.

### Analysis & changing git history.

11 Write the command to display the last five commits in repository history:

To display the last five commits in repo history you can use 'git log' command with the '-n' option to specify the no of commits to display.

```
git log -n5
```



Output :-

```
DELL@ Dell me  ~ (moon)
$ git log -n5

Commit 820445bod015f865c2395ac72e7942eocdcfa
 (HEAD → moon)
Author : sabcse < sabcse 2023@ gmail.com>
Date : Mon Feb 19' 18:23:05 2024 +0530
      Commiting program 2.

Commit e8cca898ae5ddca0836c417571 2d3c292bdac1le
Author : sabcse < sabcse 2023@ gmail.com>
Date : Mon Feb 19  13:09:51  2024 +0530

      Commiting file 6
Commit b87c4c8108603a741259cc8ed44dabbcbdf2fd1
Author : sabcse < sabcse 2023 @ gmail.com>
Date : Mon Feb 19  13:09:16 2024  +0530

      file 4.txt
Commit abe4247658f3a0f5645a012940d73b8dc0964
Author : sabcse < sabcse 2023@ gmail.com>
Date : Mon Feb 19  13:07:15  2024  +0530
      Committing file 3
```

# Program - 12

## Analysis & changing git history.

12 Write the git command to undo the changes introduced by the commit with the ID "abc123"

To undo the changes you can use the 'git revert' command here.

```
git revert abc123
```

It's important to note that 'git revert' creates a new commit that applies the inverse of the specified commit, so the original commit remains in the history.



## Output

```
DELL @ Dellme ~ (Master / REVERTING)
$ git revert -n 7a478f60fa35e2a93359297b40cbe

DELL@ Dellme ~ (Master / REVERTING)
$ git log

Commit 7a478fbe0fa35e2a93359297b40919774cbe
  (HEAD → master)
Author : sabcse <sabcse2023@gmail.com
Date : Tue Feb 20 00:50:03 2024 +0530
    Revert "message"
This reverts commit 0301d2088b17411c9826bff
                    467a4f57ca3d7987
```