Exercises for basic GIT operations:

Exercise 1: Create your first Commit.

Task:

1. Initialize your Repository: git init project Exercise-1
2. Create a file with name index.html
3. Write code in newly created file
4. Stage index.html
5. Commit with message ‘Initial Message.’

Solution:-

* git status
* git add index.html
* git commit -m "Initial Message"

Exercise 2: Commit only one file.

Task:

1. Initialize your Repository: git init Exercise-2
2. Create a files with names index.html and about.html
3. Add code in both files.
4. Stage only one file.
5. Commit with message: ‘File x added.’

Solution:-

* git status
* git add about.html
* git commit -m "File about.html added"

Exercise 3: Modify last commit.

Task:

1. Initialize your Repository: git init project Exercise-3
2. Create a file with name index.html
3. Write code in newly created file
4. Stage index.html
5. Commit with message ‘adding index.html.’
6. Add new about.html file with some code.
7. Now modify message of previous commit as ‘Adding index.html and about.html ’and add about.html file in previous commit.

Solution:-

* git status
* git add index.html
* git commit -m "adding index.html"
* git status
* git log --oneline --graph --all

\* 8c620b2 (HEAD -> master) adding index.html

* git add about.html
* git status
* git commit --amend -m "Adding index.html and about.html"
* git log --oneline --graph --all

\* d46f23f (HEAD -> master) Adding index.html and about.html

Exercise 4: Create git ignore file

Task:

1. Initialize your Repository: git init Exercise-4
2. Create a gitignore file that will ignore
   1. all files with ‘exe’ extension
   2. all files with ‘o’ extension
   3. all files with ‘jar’ extension
   4. the whole libraries directory

Solution:-

* code .
* git status
* git add .gitignore
* git commit -m "initial commit"

Exercise 5: Save your work with stash.

Task:

1. Initialize your Repository: git init Exercise-5
2. Add some random files with code
3. Now use git stash command to save required files from above files.

(For this exercise don’t send repository just submit git commands and output of **git stash list**)

Solution:-

* git status
* git add .
* git commit -m "index.html and about.html"
* code .
* git add .
* git stash

Saved working directory and index state WIP on main: c0ee410 index.html and about.html

* git stash list

stash@{0}: WIP on main: c0ee410 index.html and about.html

Exercise 6: Reset Previous Commit Keeping changes. (Soft Reset)

Task:

1. Initialize your Repository: git init Exercise-6.
2. Do some series of commits.
3. Now use git reset command to reset one of the commit and keep changes in staging area.

In answer also add output of **git status command.**

Solution:-

* git add index.html
* git commit -m "index.html file added"
* git add about.html
* git commit -m "about.html file added"
* git add contact.html
* git commit -m "contact.html file added"
* git log
* git status
* git reset --soft HEAD~1
* git status

On branch main

Changes to be committed:

(use "git restore --staged <file>..." to unstage)

new file: contact.html

* git log

Exercise 7: Reset Previous Commit (Hard Reset).

Task:

1. Initialize your Repository: git init Exercise-7.
2. Do some series of commits.
3. Use git reset command to reset one of the commit without keeping changes.

Solution:-

* git add index.html
* git commit -m "index.html file added"
* git add about.html
* git commit -m "about.html file added"
* git add contact.html
* git commit -m "contact.html file added"
* git log
* git status
* git reset --hard HEAD~1
* git status

On branch master

nothing to commit, working tree clean

Exercise 8: Merge branch (No Conflicts).

Task:

1. Initialize your Repository: git init Exercise-8
2. Create index.html file add some code.
3. Make Commit.
4. Create new branch from master branch with name my-feature.
5. Add about-us.html file with code and commit changes in my-feature branch.
6. Now checkout master branch.
7. Merge my-feature branch into master.

Solution:-

* git add index.html
* git commit -m "index.html added"
* git branch my-feature
* git branch

\* main

my-feature

* git checkout my-feature
* git add about-us.html
* git commit -m "aboutus.html added"
* git checkout main
* git merge my-feature

Exercise 9: Merge branch (Conflicts).

Task:

1. Initialize your Repository: git init Exercise-9
2. Create index.html file add some code.
3. Make Commit.
4. Create new branch from master branch with name my-feature.
5. Add about-us.html file with code
6. Make changes in index.html
7. Commit changes in my-feature branch.
8. Now checkout master branch.
9. Do some code changes in index.html
10. Commit changes.
11. Now merge my-feature branch into master.
12. You will get conflicts for index.html file. Resolve conflict by selecting both changes for the same index.html file.

Solution:-

code .

* git add index.html
* git commit -m "index.html added"
* git branch my-feature
* git branch

\* main

my-feature

* git checkout my-feature
* code .
* git status
* git add index.html
* git commit -m "index.html modified"
* git add about-us.html
* git commit -m "aboutus.html added"
* git checkout master
* code .
* git add index.html
* git commit -m "index.html file modified"
* git merge my-feature

Auto-merging index.html

CONFLICT (content): Merge conflict in index.html Automatic merge failed;

fix conflicts and then commit the result.

* git status
* git add index.html
* git status
* git commit -m "merge conflict resolved manually"

[main c15a97e] merge conflict resolved manually