FSD Laboratory 01

Name: Ansari Md Anas
Roll no: 48

Aim: Version control with Git.
Objectives:

1. To introduce the concepts and software behind version control, using the example of Git.
2. To understand the use of 'version control' in the context of a coding project.
3. To learn Git version control with Clone, commit to, and push, pull from a git repository.

Theory:
1.What is Git? What is Version Control?
Ans:

Git is a free and open-source system that provides version control. Version control essentially allows us to keep track of changes; it records what was done, who did it, why they did it (outputs your code development cycle). This enables many developers to work together on a project at the same time without conflict. This is where Git comes in as it efficiently stores the entire history of changes made to your project and lets you revert back to previous versions, track modifications that have occurred over time & help with coordinating work effectively between multiple people.

Version control is the software used to manage modifications on documents, computer programs, large web sites and so forth. It enables teams to handle the changing of source code as time goes by and they can control who commit changes, when these were committed and why those were made. This system allows people to collaborate on code, keeps track of changes made and provides a way for resolving conflictive changes made by different team members.

2. How to use Git for version controlling?
Ans:

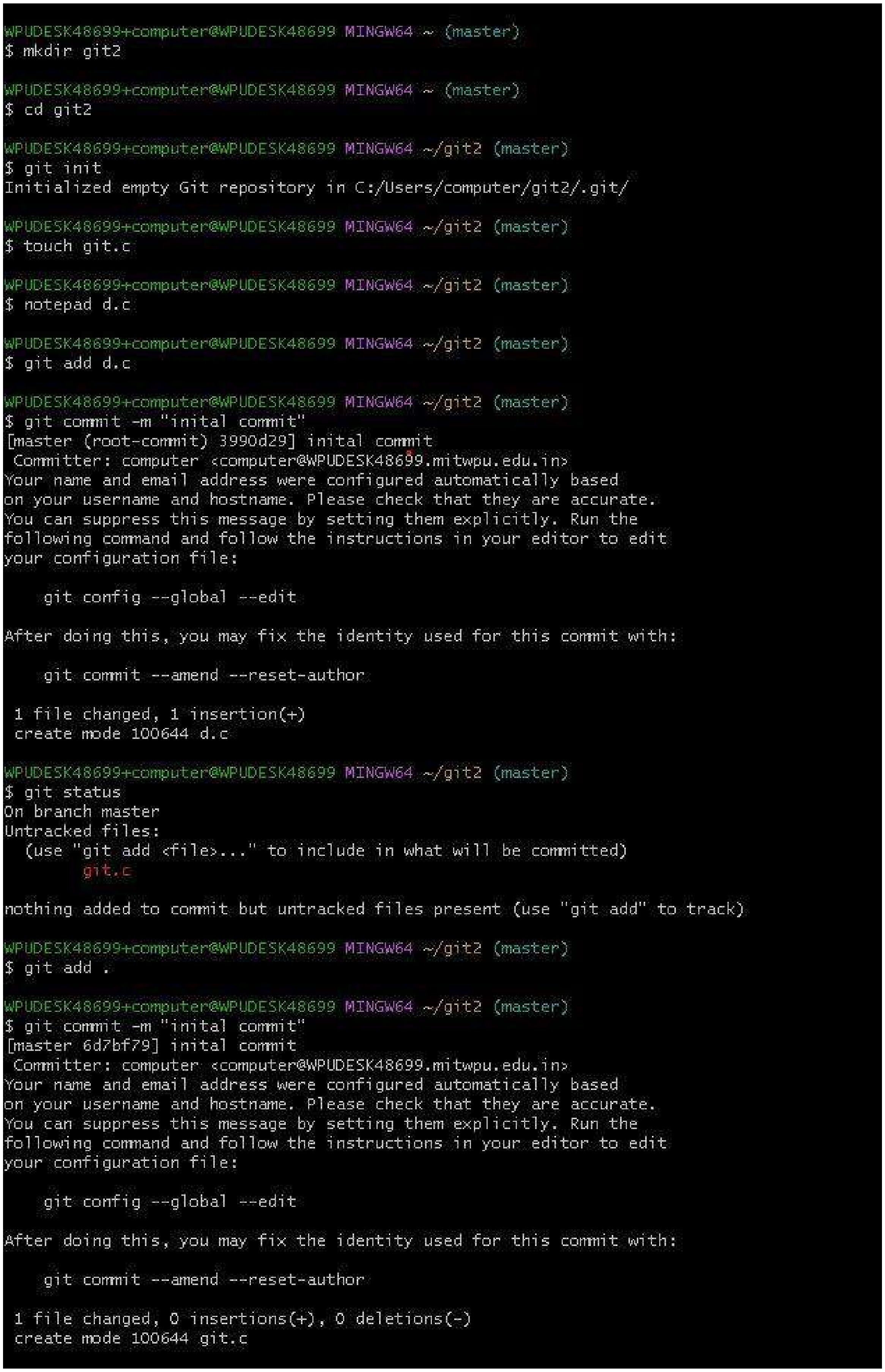
Git - Distributed Version Control System is commonly used by the software development community for maintaining source code history of any project. It permits several developers to all work on a project at the same time without taking one another changes obstructs, effectively managing entries of changes and allow users rewind their mistake from history as well as tracking modification \ effort coordinate. There is few things to do when using Git, first thing: Installing it (from the official web site), and Configuring him with your name and email. After installing Git, you may create a new repository at your project directory by executing git init. This workflow covers adding files to the staging area using git add, committing changes with git commit and pushing those changes to a remote repository such as GitHub using g One can also fetch the updates from a remote repository using git pull and perform other operations like merging branches, solving conflicts and displaying commit history with commands such as git log. So, Git is a very useful and helpful tool in source code management to share code for collaboration.

FAQ:
1.What is branching in Git?
Ans:
In Git branching is a method of development where different path ways of work are created with a repository. Each branch represents an area where you can work of features fix bugs or conduct experiments without affecting the main or master branch.
Branches facilitate collaboration and teamwork, among developers by allowing them to work on tasks concurrently in branches. A branch essentially captures the state of the code at the time it was create. Changes made in a branch can be saved without combining them with branches. This allows completed work in a branch to be merged back into the branch when ready. In case of conflicting changes Git provides solutions to help resolve these conflicts before completing the merge.
Overall utilizing branches in Git helps maintain an organized sturctures, for development effors enabling users to collaborate and manage modifications.

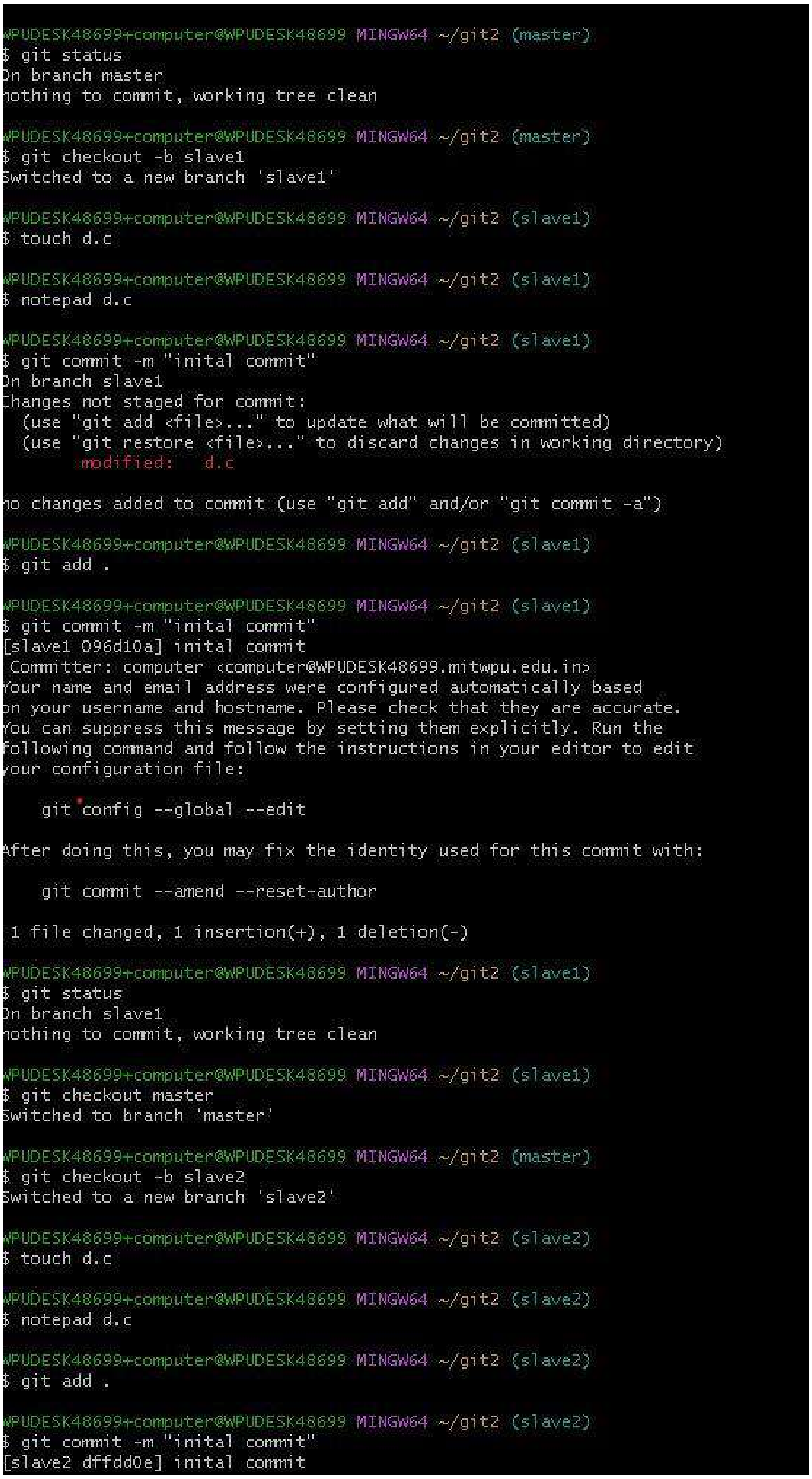
2. How to create and merge branches in Git? Write the commands used.
Ans:

To create and merge branches in Git, you start by creating a new branch using the command `git branch branch-name`, where "branch-name" is the desired name of your branch. After creating the branch, you switch to it using `git checkout branch-name`. Alternatively, you can create and switch to a new branch in one step with `git checkout -b branch-name`. Once on the new branch, you can make and commit changes independently of the main branch. When you are ready to merge the changes back into the main branch, you first switch to the main branch using `git checkout main` and then merge the feature branch with `git merge branch-name`. This process incorporates the changes from the feature branch into the main branch. If there are any conflicts, Git will prompt you to resolve them before completing the merge. This workflow allows for isolated development and seamless integration of new features or fixes into the main

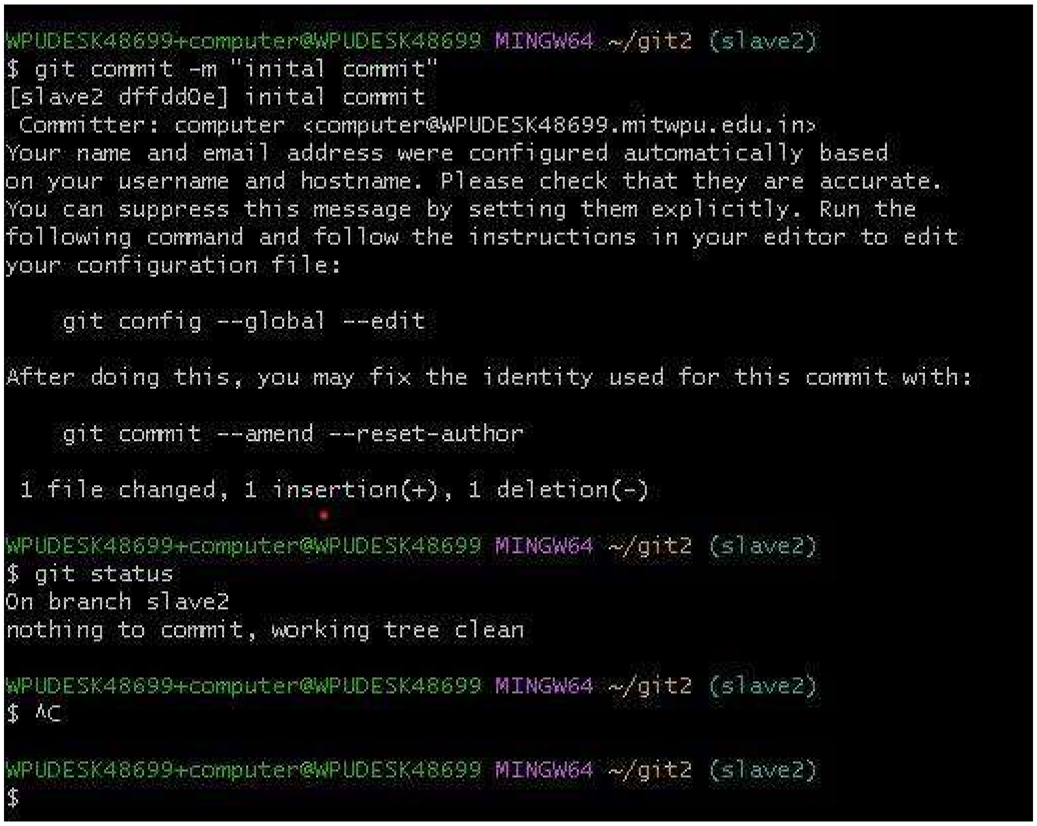
Output: Screenshots of the output to be attached.



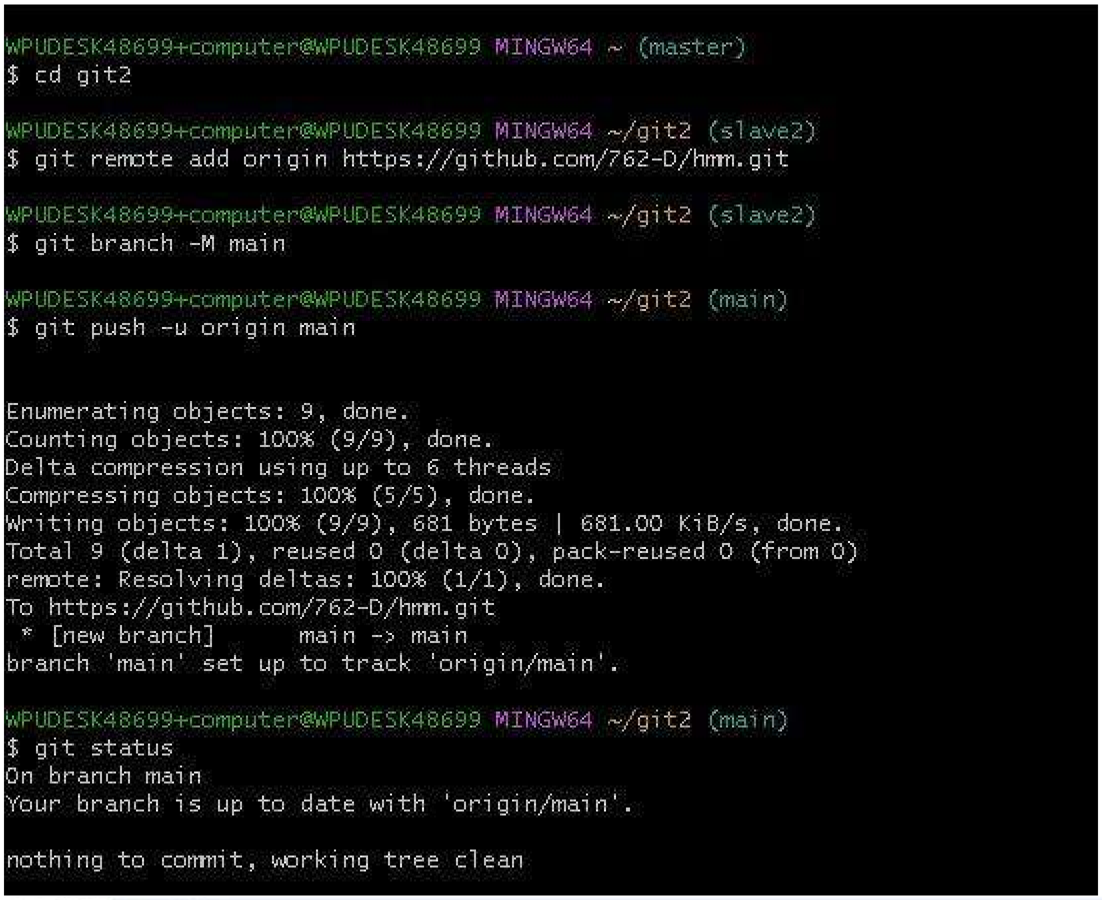
```
WPUDESK48699+computer@WPUDESK48699 MINGW64 ~ (master)
$ mkdir git2

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~ (master)
$ cd git2

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git init
Initialized empty Git repository in C:/Users/computer/git2/.git/

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ touch git.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ notepad d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git add d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git commit -m "inital commit"
[master (root-commit) 3990d29] inital commit
 Committer: computer <computer@WPUDESK48699.mitwpu.edu.in>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+)
 create mode 100644 d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        git.c

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

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git add .

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git commit -m "inital commit"
[master 6d7bf79] inital commit
 Committer: computer <computer@WPUDESK48699.mitwpu.edu.in>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

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



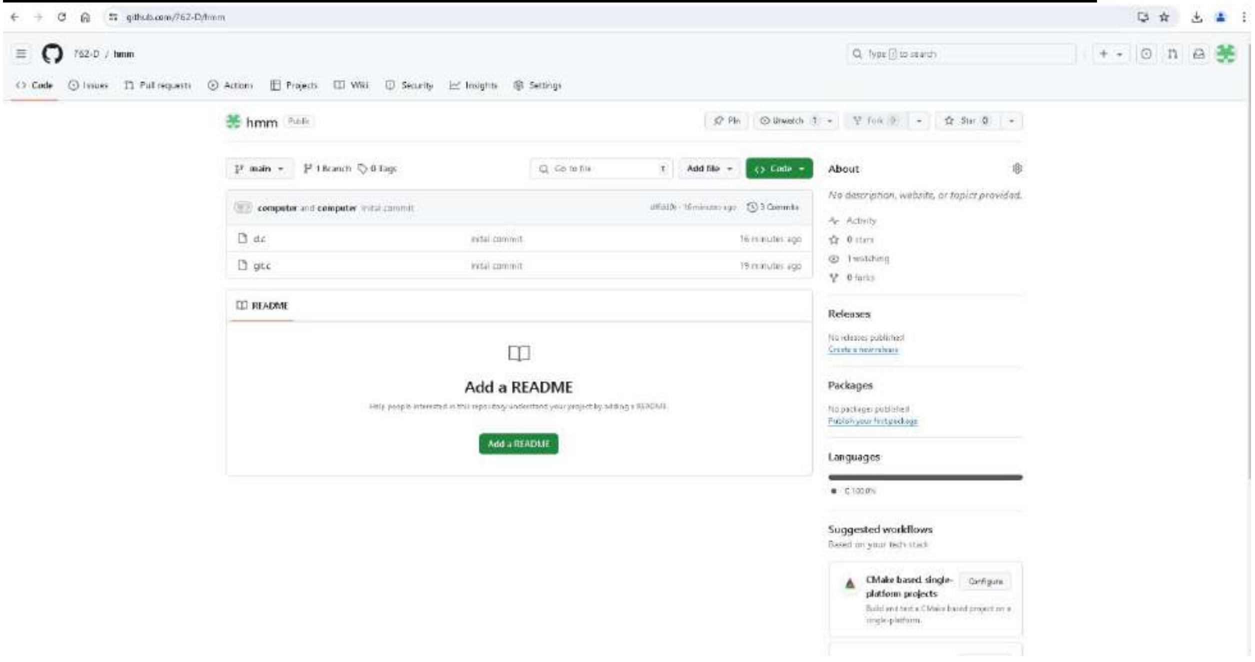
```
WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git status
On branch master
nothing to commit, working tree clean

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git checkout -b slave1
Switched to a new branch 'slave1'

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ touch d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ notepad d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ git commit -m "inital commit"
On branch slave1
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:   d.c

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

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ git add .

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ git commit -m "inital commit"
[slave1 096d10a] inital commit
 Committer: computer <computer@WPUDESK48699.mitwpu.edu.in>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+), 1 deletion(-)

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ git status
On branch slave1
nothing to commit, working tree clean

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave1)
$ git checkout master
Switched to branch 'master'

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (master)
$ git checkout -b slave2
Switched to a new branch 'slave2'

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ touch d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ notepad d.c

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git add .

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git commit -m "inital commit"
[slave2 dffdd0e] inital commit
```



```
WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git commit -m "inital commit"
[slave2 dffdd0e] inital commit
 Committer: computer <computer@WPUDESK48699.mitwpu.edu.in>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+), 1 deletion(-)

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git status
On branch slave2
nothing to commit, working tree clean

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ ^C

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$
```



```
WPUDESK48699+computer@WPUDESK48699 MINGW64 ~ (master)
$ cd git2

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git remote add origin https://github.com/762-D/hmm.git

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (slave2)
$ git branch -M main

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (main)
$ git push -u origin main


Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 6 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 681 bytes | 681.00 KiB/s, done.
Total 9 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/762-D/hmm.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

WPUDESK48699+computer@WPUDESK48699 MINGW64 ~/git2 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```



github.com/762-D/hmm

762-D / hmm

hmm Public

main · 1 Branch · 0 Tags

computer and computer initial commit · 3 Commits

d.c	initial commit	16 minutes ago
git.c	initial commit	19 minutes ago

README

Add a README

Help people interested in this repository understand your project by adding a README.

About

No description, website, or topics provided.

0 stars · 1 watching · 0 forks

Releases: No releases published

Packages: No packages published

Languages: C 100.0%

Suggested workflows: CMake based, single-platform projects — Build and test a CMake based project on a single-platform.

Problem Statement:

Create a public git repository for your team and submit the repo URL as a solution to this assignment, Learn Git concept of Local and Remote Repository, Push, Pull, Merge and Branch.