Project 0

Thursday, April 6, 2023 2:28 PM

To Be Done On Windows (Maedeh)

```
Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop
$ mkdir project0

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop
$ cd project0/

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0
$ ls --all
./  ../

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0
$ git status
fatal: not a git repository (or any of the parent directories): .git

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0
$ git init
Initialized empty Git repository in C:/Users/Asus/Desktop/project0/.git/

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0
$ ls --all
./  ../ .git/

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ touch a.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ echo hello >> a.txt
```

WHAT IS THE STATE OF A.TXT RIGHT NOW?

```
Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a.txt

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

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git add a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it
```

WHAT IS THE STATE OF A.TXT RIGHT NOW?

```
Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   a.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git commit -m "created a.txt"
[master (root-commit) 95d7310] created a.txt
 1 file changed, 1 insertion(+)
 create mode 100644 a.txt
```

WHAT IS THE STATE OF A.TXT RIGHT NOW?

```
Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master
nothing to commit, working tree clean

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git log
commit 95d7310542bda98d9f4a0ba21f8129b6bceb857e (HEAD -> master)
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date:   Thu Apr 6 13:36:27 2023 -0700

    created a.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ echo bye >> a.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (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:   a.txt

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

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ touch b.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ echo "hello b" >> b.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (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:   a.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        b.txt

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

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git add .
```

```
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'b.txt', LF will be replaced by CRLF the next time Git touches it

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   a.txt
        new file:   b.txt


Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git commit -m "created b.txt and modified a.txt"
[master 8985b9e] created b.txt and modified a.txt
 2 files changed, 2 insertions(+)
 create mode 100644 b.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git log
commit 8985b9e04ae023362765806874538853632673 6e (HEAD -> master)
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date:   Thu Apr 6 13:38:57 2023 -0700

    created b.txt and modified a.txt

commit 95d7310542bda98d9f4a0ba21f8129b6bceb857e
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date:   Thu Apr 6 13:36:27 2023 -0700

    created a.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ mkdir dir

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master
nothing to commit, working tree clean

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ touch dir/c.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ echo "hello c" >> dir/c.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dir/

nothing added to commit but untracked files present (use "git add" to track)
Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git add .
warning: in the working copy of 'dir/c.txt', LF will be replaced by CRLF the next time Git touches it

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git commit -m "created dir/c.txt"
[master 2e35a02] created dir/c.txt
 1 file changed, 1 insertion(+)
 create mode 100644 dir/c.txt

Asus@DESKTOP-UNNUFAI MINGW64 ~/Desktop/project0 (master)
$ git log
commit 2e35a0247113c913007e0c08d75f44d266c96fc3 (HEAD -> master)
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date:   Thu Apr 6 13:40:46 2023 -0700
```

created dir/c.txt

commit 8985b9e04ae023362765806874538853632673 6e
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date: Thu Apr 6 13:38:57 2023 -0700

created b.txt and modified a.txt

commit 95d7310542bda98d9f4a0ba21f8129b6bceb857e
Author: Maedeh Heydari <maedeh.heydari212@gmail.com>
Date: Thu Apr 6 13:36:27 2023 -0700

created a.txt

From <https://mail.google.com/mail/u/0/#inbox/FMfcgzGsltMtjGtMjJncTlvbzhXDxzMR>

Project 1

Saturday, February 11, 2023 6:50 PM

To Be Done On Windows

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ touch first.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ echo "hello" >> first.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ touch second.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ echo "hello2" >> second.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ git status
fatal: not a git repository (or any of the parent directories): .git

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1
$ git init
Initialized empty Git repository in C:/Users/Logan/Desktop/College/Git Workshop/Project 1/.git/
```

! BEFORE MOVING ON TO THE NEXT COMMAND:
Try to guess what state first and second.txt are in.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt
        second.txt

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

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git add second.txt
warning: LF will be replaced by CRLF in second.txt.
The file will have its original line endings in your working directory
```

! BEFORE MOVING ON TO THE NEXT COMMAND:

Try to guess what state first and second.txt are in.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   second.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt


Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git commit -m "commit second.txt"
[master (root-commit) 3a324a5] commit second.txt
 1 file changed, 1 insertion(+)
 create mode 100644 second.txt
```

! BEFORE MOVING ON TO THE NEXT COMMAND:
Try to guess what state first and second.txt are in.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt

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

- **Check commit history using git log note how your registered name and email are designated as the author.**
- **Note that git status does not list unchanged (commited) files.**

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ echo "edit" >> second.txt
```

! BEFORE MOVING ON TO THE NEXT COMMAND:
Try to guess what state first and second.txt are in.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (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:   second.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        first.txt

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

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git add .
warning: LF will be replaced by CRLF in second.txt.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in first.txt.
The file will have its original line endings in your working directory
```

! BEFORE MOVING ON TO THE NEXT COMMAND:
Try to guess what state first and second.txt are in.
Then run git status and check your guess

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git commit -m "edit second.txt and commit first.txt"
[master 982d507] edit second.txt and commit first.txt
 2 files changed, 2 insertions(+)
 create mode 100644 first.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ rm second.txt
```

! BEFORE MOVING ON TO THE NEXT COMMAND:
Try to guess what state first and second.txt are in.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    second.txt

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

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git add second.txt
```

- It may seem counterintuitive that `git add` is used to stage deletion. But deletion is treated almost exactly like simple modification. As the git status message suggests, `git rm` can also be used to stage deletion.

- Note git messages. A lot of times they suggest just the command you might be looking for.

```
Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    second.txt


Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ echo "edit1" >> first.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    second.txt

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:   first.txt

Logan@DESKTOP-Q0U5DEF MINGW64 ~/Desktop/College/Git Workshop/Project 1 (master)
$ git commit
[master bfe022f] my message
 1 file changed, 1 insertion(+)
 create mode 100644 second.txt
```

- Learn how to create a short commit message and also include a longer description
- Check with git log and git log --oneline what the difference is.

Project 2

Saturday, February 11, 2023 9:25 PM

To Be Done On Linux/WSL

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git init
Initialized empty Git repository in /mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2/.git/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add --all
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create first and second files"
[master (root-commit) 3b20e06] create first and second files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
 create mode 100644 second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ mkdir hello
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch hello/third.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello/

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add hello/third.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello/third.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git restore --staged hello/third.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello/

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add hello/*
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello/third.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create third file and hello dir"
[master d6f3470] create third file and hello dir
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 hello/third.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "edit" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   first.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git restore first.txt
```

★ BEFORE MOVING ON:
TRY AND GUESS first.txt 's content at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
edit
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "add a line to first.txt"
[master 8f0b61b] add a line to first.txt
 1 file changed, 1 insertion(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "second edit" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git restore *
```

★ BEFORE MOVING ON:

TRY AND GUESS first.txt 's content at this point

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
Edit

Commit changes so far with msg add a line to first.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
8f0b61b (HEAD -> master) add a line to first.txt
d6f3470 create third file and hello dir
3b20e06 create first and second files
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git revert 8f0b61b
[master cf97f41] Revert "add a line to first.txt"
 1 file changed, 1 deletion(-)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
cf97f41 (HEAD -> master) Revert "add a line to first.txt"
8f0b61b add a line to first.txt
d6f3470 create third file and hello dir
3b20e06 create first and second files
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "first edit" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
first edit
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m "add something to first.txt"
[master 9716028] add something to first.txt
 1 file changed, 1 insertion(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "second edit" >> second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m "add something to second.txt"
[master 7bec684] add something to second.txt
 1 file changed, 1 insertion(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
7bec684 (HEAD -> master) add something to second.txt
9716028 add something to first.txt
cf97f41 Revert "add a line to first.txt"
8f0b61b add a line to first.txt
d6f3470 create third file and hello dir
3b20e06 create first and second files
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git revert 9716028
[master 47f4101] Revert "add something to first.txt"
 1 file changed, 1 deletion(-)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
47f4101 (HEAD -> master) Revert "add something to first.txt"
7bec684 add something to second.txt
9716028 add something to first.txt
cf97f41 Revert "add a line to first.txt"
8f0b61b add a line to first.txt
d6f3470 create third file and hello dir
3b20e06 create first and second files

BEFORE MOVING ON:
TRY AND GUESS second.txt 's content at this point

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat second.txt
second edit

Also cat first.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "reset first" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "reset second" >> second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m "make changes to first and second.txt"
[master 1d0a871] make changes to first and second.txt
 2 files changed, 2 insertions(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "reset first 1" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "reset second 2" >> second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
reset first
reset first 1
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat second.txt
second edit
reset second
reset second 2

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   first.txt

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:   second.txt

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
1d0a871 (HEAD -> master) make changes to first and second.txt
47f4101 Revert "add something to first.txt"
7bec684 add something to second.txt
9716028 add something to first.txt
cf97f41 Revert "add a line to first.txt"
8f0b61b add a line to first.txt
d6f3470 create third file and hello dir
3b20e06 create first and second files
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git reset 47f4101
Unstaged changes after reset:
M       first.txt
M       second.txt
```

BEFORE MOVING ON:
TRY AND GUESS first and second.txt 's content AND state (commited, staged, etc?)

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ 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:   first.txt
        modified:   second.txt

no changes added to commit (use "git add" and/or "git commit -a")
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
reset first
reset first 1
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat second.txt
second edit
reset second
reset second 2
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git diff first.txt
diff --git a/first.txt b/first.txt
index e69de29..f9f473b 100644
--- a/first.txt
+++ b/first.txt
@@ -0,0 +1,2 @@
+reset first
+reset first 1
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git diff second.txt
diff --git a/second.txt b/second.txt
index 465310c..0dcef9a 100644
--- a/second.txt
+++ b/second.txt
@@ -1 +1,3 @@
 second edit
+reset second
+reset second 2


kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ rm -rf *
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ rm -rf .git
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git init
Initialized empty Git repository in /mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2/.git/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add --all
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create first and second text files"
[master (root-commit) 4b099ee] create first and second text files
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
 create mode 100644 second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "hello first" >> first.txt
```

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m "put some text into first.txt"
[master 683ad46] put some text into first.txt
 1 file changed, 1 insertion(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "hello second" >> second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ 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:   second.txt

no changes added to commit (use "git add" and/or "git commit -a")
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit --amend --no-edit
[master 881cd52] put some text into first.txt
 Date: Sun Feb 12 00:18:08 2023 +0330
 2 files changed, 2 insertions(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
nothing to commit, working tree clean
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat second.txt
hello second
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
881cd52 (HEAD -> master) put some text into first.txt
4b099ee create first and second text files
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git diff 881cd52 4b099ee
diff --git a/first.txt b/first.txt
index 9622cbc..e69de29 100644
--- a/first.txt
+++ b/first.txt
@@ -1 +0,0 @@
-hello first
diff --git a/second.txt b/second.txt
index f46daac..e69de29 100644
--- a/second.txt
+++ b/second.txt
@@ -1 +0,0 @@
-hello second

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit --amend -m "new message!"
[master ffe825c] new message!
 Date: Sun Feb 12 00:18:08 2023 +0330
 2 files changed, 2 insertions(+)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
ffe825c (HEAD -> master) new message!
4b099ee create first and second text files
```

```
git init
touch first.txt
touch second.txt
git add --all
git commit -m "create first and second files"
mkdir hello
touch hello/third.txt
git status
git add hello/third.txt
git status
git restore --staged hello/third.txt
git status
git add hello/*
git status
git commit -m "create third file and hello dir"
echo "edit" >> first.txt
git add first.txt
git status
git restore first.txt
cat first.txt
git status
git commit -m "add a line to first.txt"
echo "second edit" >> first.txt
git restore *
cat first.txt
git log --oneline
git revert 8f0b61b
```

```
git log --oneline
cat first.txt
echo "first edit" >> first.txt
cat first.txt
git commit -a -m "add something to first.txt"
echo "second edit" >> second.txt
git commit -a -m "add something to second.txt"
git log --oneline
git revert 9716028
git log --oneline
cat second.txt
echo "reset first" >> first.txt
echo "reset second" >> second.txt
git commit -a -m "make changes to first and second.txt"
echo "reset first 1" >> first.txt
echo "reset second 2" >> second.txt
cat first.txt
cat second.txt
git add first.txt
git status
git log --oneline
git reset 47f4101
git status
cat first.txt
cat second.txt
git diff first.txt
git diff second.txt
rm -rf *
rm -rf .git
git init
touch first.txt
touch second.txt
git add --all
git commit -m "create first and second text files"
echo "hello first" >> first.txt
git commit -a -m "put some text into first.txt"
echo "hello second" >> second.txt
git status
git add second.txt
git commit --amend --no-edit
git status
cat second.txt
git log --oneline
git diff 881cd52 4b099ee
git commit --amend -m "new message!"
git log --oneline
```

Project 3

Sunday, February 12, 2023 12:53 AM

To Be Done On Linux/WSL

code

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git init
Initialized empty Git repository in /mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2/.git/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        code.c

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add .
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create
code file"
[master (root-commit) 2e41501] create code file
 1 file changed, 13 insertions(+)
 create mode 100644 code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ nano .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat .gitignore
.gitignore
hello/
a*

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ gcc code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
nothing to commit, working tree clean
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ mkdir hello
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ ./a.out
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
nothing to commit, working tree clean
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag -a v.1.0 -m "first
tag"
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag
v.1.0
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git show v.1.0
tag v.1.0
Tagger: Amir Kooshky <kooshkya@gmail.com>
Date:   Sun Feb 12 00:57:59 2023 +0330

first tag

commit 2e4150165969dd87f9101709785e094983de5023 (HEAD -> master, tag: v.1.0)
Author: Amir Kooshky <kooshkya@gmail.com>
Date:   Sun Feb 12 00:55:26 2023 +0330

    create code file

diff --git a/code.c b/code.c
new file mode 100644
index 0000000..90e3e3e
--- /dev/null
+++ b/code.c
@@ -0,0 +1,13 @@
+#include <stdio.h>^M
+int main() {^M
+       int i;^M
+       FILE * fptr;^M
+       char fn[50];^M
+       char str[] = "Git Tutorial\n";^M
+       fptr = fopen("hello/test.txt", "w");^M
+       for (i = 0; str[i] != '\n'; i++) {^M
+           fputc(str[i], fptr);^M
```

```
+       }^M
+       fclose(fptr);^M
+       return 0;^M
+}
\ No newline at end of file
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag -d v.1.0
Deleted tag 'v.1.0' (was f832f95)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag
```

Some more practice with .gitignore

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git init
Initialized empty Git repository in /mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2/.git/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ nano code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat code.c
#include <stdio.h>
int main() {
      int i;
      FILE * fptr;
      char fn[50];
      char str[] = "Git Tutorial\n";
      fptr = fopen("hello/test.txt", "w");
      for (i = 0; str[i] != '\n'; i++) {
         fputc(str[i], fptr);
      }
      fclose(fptr);
      return 0;
}
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create
code file"
[master (root-commit) 13fee82] create code file
 1 file changed, 13 insertions(+)
 create mode 100644 code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ gcc code.c
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      a.out

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo
"*.out" >> .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      .gitignore

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo
".gitignore" >> .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
nothing to commit, working tree clean
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ mkdir hello/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ mkdir a
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ mkdir a/hello
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch a/hello/my.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
      a/

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ ./a.out
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
        a/
        hello/

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo
"hello/" >> .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
nothing to commit, working tree clean
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat .gitignore
*.out
.gitignore
hello/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ nano .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat .gitignore
*.out
.gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a/
        hello/

nothing added to commit but untracked files present (use "git add" to track)
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo
"/hello/" >> .gitignore
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        a/

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

```
git init
git status
git add .
git commit -m "create code file"
touch .gitignore
nano .gitignore
cat .gitignore
gcc code.c
git status
mkdir hello
./a.out
git status
git tag
git tag -a v.1.0 -m "first tag"
git tag
git show v.1.0
git tag -d v.1.0
git tag
git init
touch code.c
nano code.c
cat code.c
git add code.c
git commit -m "create code file"
gcc code.c
git status
touch .gitignore
echo "*.out" >> .gitignore
git status
echo ".gitignore" >> .gitignore
git status
gcc code.c -out my.out
./a.out
mkdir hello/
mkdir a
mkdir a/hello
```

```
touch a/hello/my.txt
git status
./a.out
git status
echo "hello/" >> .gitignore
git status
cat .gitignore
nano .gitignore
cat .gitignore
git status
echo "/hello/" >> .gitignore
git status
```

Project 4

Sunday, February 12, 2023 4:50 PM

To Be Done On Linux/WSL

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git init
Initialized empty Git repository in /mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2/.git/
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add .
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create
first.txt"
[master (root-commit) af86102] create first.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
```

★ Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch
* master
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
af86102 (HEAD -> master) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch firstBranch
```

★ Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch
  firstBranch
* master
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
af86102 (HEAD -> master, firstBranch) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell
```

★ Before Moving On!
Try and guess how the DAG would have been different if we had used
git checkout -b firstBranch instead of git branch firstBranch

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git tag firstCommitTag
```

★ Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
af86102 (HEAD -> master, tag: firstCommitTag, firstBranch) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch second.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "first edit" >> first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add .
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "create
second.txt and edit first.txt"
[master b5265b1] create second.txt and edit first.txt
 2 files changed, 1 insertion(+)
 create mode 100644 second.txt
```

★ Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
b5265b1 (HEAD -> master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag, firstBranch) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout firstBranch
Switched to branch 'firstBranch'
```

★ Before Moving On!
Try and draw the git repo's DAG at this point. Also, try to guess what the working directory and the index look like.

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
af86102 (HEAD -> firstBranch, tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
b5265b1 (master) create second.txt and edit first.txt
af86102 (HEAD -> firstBranch, tag: firstCommitTag) create first.txt
```

★ Note the difference between normal git log and git log --all

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ ls
first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "another edit" >>
first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ cat first.txt
another edit
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m "firstBranch:
Do another edit to first.txt"
[firstBranch f725f4f] firstBranch: Do another edit to first.txt
```

1 file changed, 1 insertion(+)

★ Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
f725f4f (HEAD -> firstBranch) firstBranch: Do another edit to first.txt
b5265b1 (master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout firstCommitTag
Note: switching to 'firstCommitTag'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at af86102 create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout firstBranch
Previous HEAD position was af86102 create first.txt
Switched to branch 'firstBranch'
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
f725f4f (HEAD -> firstBranch) firstBranch: Do another edit to first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout af86102
Note: switching to 'af86102'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at af86102 create first.txt
```

★ Note that checking out a tag and checking out a commit had the same effect: Both detached the HEAD

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ touch detached.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git add .
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -m "detached:
created a file"
```

```
[detached HEAD 1bfe63f] detached: created a file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 detached.txt
```

Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout master
Warning: you are leaving 1 commit behind, not connected to
any of your branches:

  1bfe63f detached: created a file

If you want to keep it by creating a new branch, this may be a good time
to do so with:

  git branch <new-branch-name> 1bfe63f

Switched to branch 'master'
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
f725f4f (firstBranch) firstBranch: Do another edit to first.txt
b5265b1 (HEAD -> master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ check git graph

Command 'check' not found, but can be installed with:

sudo apt install gitlab-shell

kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "Our Commit is Lost!"
Our Commit is Lost!
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout firstBranch
Switched to branch 'firstBranch'
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
f725f4f (HEAD -> firstBranch) firstBranch: Do another edit to first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout -b secondBranch
Switched to a new branch 'secondBranch'
```

Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline
f725f4f (HEAD -> secondBranch, firstBranch) firstBranch: Do another edit to first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "another change" >>
first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m
"secondBranch: create some changes to first.txt"
[secondBranch b0a5db1] secondBranch: create some changes to first.txt
 1 file changed, 1 insertion(+)
```

Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
b0a5db1 (HEAD -> secondBranch) secondBranch: create some changes to first.txt
f725f4f (firstBranch) firstBranch: Do another edit to first.txt
b5265b1 (master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch -d firstBranch
Deleted branch firstBranch (was f725f4f).
```

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
b0a5db1 (HEAD -> secondBranch) secondBranch: create some changes to first.txt
f725f4f firstBranch: Do another edit to first.txt
b5265b1 (master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git branch thirdBranch
```

Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "second branch edit" >>
first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git commit -a -m
"secondBranch: make an edit to first.txt"
[secondBranch 467605c] secondBranch: make an edit to first.txt
 1 file changed, 1 insertion(+)
```

Before Moving On!
Try and draw the git repo's DAG at this point

```
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
467605c (HEAD -> secondBranch) secondBranch: make an edit to first.txt
b0a5db1 (thirdBranch) secondBranch: create some changes to first.txt
f725f4f firstBranch: Do another edit to first.txt
b5265b1 (master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout b0a5db1
```

Before Moving On!
Try and guess the state of the head after this instruction

```
Note: switching to 'b0a5db1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at b0a5db1 secondBranch: create some changes to first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git log --oneline --all
467605c (secondBranch) secondBranch: make an edit to first.txt
b0a5db1 (HEAD, thirdBranch) secondBranch: create some changes to first.txt
f725f4f firstBranch: Do another edit to first.txt
b5265b1 (master) create second.txt and edit first.txt
af86102 (tag: firstCommitTag) create first.txt
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ echo "check git graph"
check git graph
kooshkya@DESKTOP-Q0U5DEF:/mnt/c/Users/Logan/Desktop/College/Git Workshop/Project 2$ git checkout thirdBranch
Switched to branch 'thirdBranch'
```

Project 5

Sunday, February 12, 2023 5:52 PM

To Be Done On Intellij

Start an intellj empty project
Git init
Show git tab
Create .gitignore from editor file explorer
Put .idea and .gitignore into .gitignore
Show commit tab (left side of editor)
Create a main file with this content (simple hello world):

Main

Show commit tab
Commit from commit tab
Show git tab => show tags and branches and head
Show commit diff on git tab
Create new branch called "feature" and checkout to it (separately)
Show git tab
Put a new line into Main.java => show how diffs are checked in the editor
Show how diffs can be reversed in the editor
Create a new class called Help.java
Commit from commit tab
Show git tab
Checkout main/master and show differences
Show git tab and where head is.
Checkout -b to a new branch feature2
Put a new line into Main.java
Create a new file called feature2.txt
Commit
Show git tab
Regret feature2
Checkout master/main
Create feature2Debug
Make other changes to Main and create feature2.java
Show git tab
Checkout main/master
Checkout -b feature4
Create feature.java and commit
Modify feature.java and commit
Regret modifying feature
Git checkout -b feature4Debug to create feature.java by finding its hash in git tab
Show git tab
Make a different change to feature.java and commit
Show git tab

Project 5.1.

Sunday, February 12, 2023 6:44 PM

To Be Done On Intellij

An extension of project 5

We now wish to "**merge feature INTO master"**

BEFORE MOVING ON
TRY TO GUESS WHAT COMMAND WE SHOULD BE USING

Git checkout master
Show git tab
Git merge feature

BEFORE MOVING ON
TRY TO GUESS WHAT TYPE OF MERGE HAS JUST HAPPENED

Note the git message for the merge

Check git tab

Check files for changes (check each line with intellij git options)

Project 5.2.

Sunday, February 12, 2023 7:05 PM

To Be Done On Intellij

An extension of project 5

Git checkout feature2Debug
Remove change to Main.java
Commit
Git checkout master
Check git tab => note where head, master, and feature2Debug are!
Git merge feature2Debug

BEFORE MOVING ON
TRY TO GUESS WHAT THE REPO DAG LOOKS LIKE

Check git tab => note where head, master, and feature2Debug are!

Project 6

Sunday, February 12, 2023 7:42 PM

To Be Done On Intellij, VSCode, and a plain text editor like notepad

Create a java project on intellij and start its git repo
Create Main.java and commit
Checkout to new branch feature1
Make a change to a line in Main.java
Commit
Checkout master
Checkout new branch feature2
Make a change in the same line in Main.java
Commit the change
Checkout master
Merge feature1 into master

- What kind of merge is used here?

Merge feature2 into master
Show people the git message
Open main.java on notepad (or simple intellij) and show git notation
Open main.java on VSCode and show VSCode notation
Open Intellij conflict resolution tab and explain how it works
Resolve conflicts using all three (and git merge --abort afterwards to redo it again)
Git commit the last conflic resolved version
Check git tab

Project 6.1.

Sunday, February 12, 2023 8:10 PM

To Be Done On Intellij

Create a java project and its git repo
Create Main.java and put main in it and commit
Checkout to branch feature1
Create a new function feature1 ABOVE main and commit
Checkout to master
Checkout to branch feature2
Create a new function feature2 BELOW main and commit
Checkout to master
Put a print statement into main.java and commit
Merge feature1
Check git tab
Merge feature2
Check git tab
See? No conflict the first time!
Sove second merge's conflict on intellij

Project 7

Sunday, February 12, 2023 9:02 PM

To Be Done On VSCode

Try to clone https://github.com/kooshkya/Project7
Use WSL terminal (without SSH access) to clone and see error
Use WSL terminal to clone using https and see it's ok
Explain the access difference
Teach how to start ssh on wsl (and on linux) from https://docs.github.com/en/authentication/connecting-to-github-with-ssh/about-ssh and emphasize importance of the link
Clone again with SSH and show that it is possible.
Open git graph and show that branches have been downloaded. Note that the master branch is the only branch with both a local and an origin version (show with both git graph AND git log --all). Explain the difference between origin and local branches(use the next slide).
Check out readme and show how a local version is created (using git log)
Put some text in the readme file and commit
Show git graph and git log
Checkout origin/readme and show that it results in a detached head
Checkout master again
Download the version (tag specified) from git hub and git status on it to show it is not a repo!

Project 7.1.

Sunday, February 12, 2023 9:41 PM

To Be Done On VSCode

Go to GitHub for project 7 at https://github.com/kooshkya/Project7 and make a change to Main.java (create a comment or called GitHub change.)
Then make a change (add GitHub change) to readme.md ON THE README branch
Go back to the local repo and show the git graph and git log
Git fetch origin on the local repo and show the changes.
Checkout origin/readme and show the readme file and note that all new commits have been downloaded from origin
Checkout origin/master and show the contents of Main.java
Checkout your own master and readme branches and show that they are unchanged.

Clone the repo again in a different directory and show the git graph and point out the differences.

Project 7.2.

Thursday, February 16, 2023 12:18 PM

To Be Done On VSCode

Update master on GitHub: add a comment to the top of Main.java
Go back to local repo and show by git log and git graph that origin/master has not been updated and the new commit has not been downloaded.
Git fetch and show how the new commit has been downloaded and how origin/main is now pointing to it.
Checkout master and origin/master and show the contents of Main.java in each
Checkout master and git pull origin master
Explain how a fast forward merge has happened. Show in Git Graph and Git Log how local master has moved. Show Main.java in local master
Go to GitHub and make a change to readme.md on the readme branch that makes sure it conflicts with the changes on the local readme branch.
Go to local and show how origin/readme is not yet updated.
Checkout local readme and show the contents of readme.md here.
Git pull origin readme
Note how origin/readme is updated and its new commit is downloaded -> git fetch has been executed.
Explain how this is a conflicted three-headed merge. Explain how origin/readme can be regarded as a local branch
Resolve the conflicts and commit them --no-edit

Project 7.3.

Thursday, February 16, 2023 11:28 AM

To Be Done On VSCode

Start from the local repo you made in 7.2.
Git fetch and checkout featureBranch branch and note it is in full sync with origin/featureBranch. Also note how the local version was just created while it didn't exist before.
Make a change to Main.java's feature function and commit
Show git graph and git log and note how local featureBranch has advanced by one commit but origin/featureBranch has not changed.
Go to GitHub and show how Main.java is unchanged on the featureBranch branch. Also show that the commit you just made does not appear on its commit list.
On local, git checkout origin/featureBranch and show that it matches GitHub.
Git push origin featureBranch and explain how a fast forward merge takes place.
Show git graph and git log and show how origin/featureBranch has moved.
Go to GitHub and show Main.java and how it has changed Also show how the new commit has been uploaded.
Checkout master and create a new branch named testBranch and check it out
Create test.txt with some text inside and commit to testBranch
Show using git graph, git log, and GitHub that origin/testBranch is nonexistent
Git push test branch and show how an origin/testBranch is created by using both git graph, git log, and GitHub
Make a change to test.txt on GitHub (put text on the second line) and commit.
Go to local and show how origin/test has not changed and the new commit does not appear.
Make a change to test.txt on local so it clashes with that made on GitHub and commit.
Show in GitHub that the local change is unnoticed by origin.
Git push origin testBranch and show by the error that git fetch has NOT occurred.
Git fetch and show how the GitHub changes are imported.
Git push origin testBranch again.
Explain why push is not possible due to a non-fast-forward error.
Explain how this can be resolved by pulling
Git pull origin testBranch
Resolve conflicts and commit
Show git graph and git log and explain
Now push origin testBranch and explain this is possible because it follows a fast-forward merge paradigm.
Show git graph and git log and explain what has happened.