PS 3 (George Wang)

SOLO

1. Late coins used this pset: ***0***
2. Late coins left after submission: ***4***

This submission is my work alone and complies with the 30538 integrity policy." Add your initials to indicate your agreement: ***GW***

I have uploaded the names of anyone I worked with on the problem set here" ***NA***

Learn git branching (15 points)

Go to https://learngitbranching.js.org. This is the best visual git explainer we know of.

1. Complete all the levels of main "Introduction Sequence". Report the commands needed to complete "Git rebase" with one line per command.

```
git branch bugfix
git checkout bugfix
git commit -m 'bugfix'
git checkout main
git commit -m 'main'
git checkout bugfix
git rebase main
```

2. Complete all the levels of main "Ramping up". Report the commands needed to complete "Reversing changes in git" with one line per command.

```
git reset C1
git checkout pushed
git revert pushed
```

3. Complete all the levels of remote "Push & Pull – Git Remotes!". Report the commands needed to complete "Locked Main" with one line per command.

```
git branch feature
git push origin feature
git reset HEAD~1
git checkout feature
git push origin feature
```

Exercises

- Basic Staging and Branching (10-15)

1. Exercise. For your pset submission, tell us only the answer to the last question (22).

```
On branch master
nothing to commit, working tree clean

It means that there are no changes in the working directory, as the file has
been restored to the last commit.
```

2. Exercise. For your pset submission, tell us only the output to the last question (18).

```
(base) MBP14inch2066:exercise georgew$ git diff mybranch master
diff --git a/file1.txt b/file1.txt
deleted file mode 100644
index 4f29c7a..0000000
--- a/file1.txt
+++ /dev/null
@@ -1 +0,0 @@
-George Wang
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..4ab7e6d
--- /dev/null
+++ b/file2.txt
@@ -0,0 +1 @@
+This is file 2

Two branches have diverged and contain different files. mybranch has
file1.txt while master has file2.txt, which explains why file2.txt is not
visible in the previous directory.
```

- Merging

1. Exercise. After completing all the steps (1 through 12), run git log –oneline –graph –all and report the output.

```
(base) MBP14inch2066:exercise georgew$ git log --oneline --graph --all
* ba2cc21 (HEAD -> master) Change greeting to uppercase
* ee567d4 Add content to greeting.txt
* 2cffaa2 Add file greeting.txt
```

2. Exercise. Report the answer to step 11.

```
(base) MBP14inch2066:exercise georgew$ git commit -m "Merge branch
'greeting'"
[master 76adbb2] Merge branch 'greeting'
(base) MBP14inch2066:exercise georgew$ git log --oneline --graph --all
*   76adbb2 (HEAD -> master) Merge branch 'greeting'
|\
| * 5111d5f (greeting) Update greeting.txt with my favorite greeting
* | 0a89e72 Add README.md with repository information
|/
* 1d2227f Add content to greeting.txt
* 984689d Add file greeting.txt
(base) MBP14inch2066:exercise georgew$
```

3. Identify the type of merge used in Q1 and Q2 of this exercise. In words, explain the difference between the two merge types, and describe scenarios where each type would be most appropriate.

Q1: In the first exercise fast foward merge, when I created the branch (feature/uppercase) and made changes to it, Git performed a fast-forward merge. It occurs when the master branch hasn't diverged from the feature/uppercase branch. It doesn't create a merge because changes of the feature branch are ahead of the master branch.

Q2: In the second exercise 3-Way Merge, I created two branches (greeting and master) and made independent changes to each. When I merged greeting into master, Git performed a 3-way merge because branches had diverged from their common ancestor. This type of merge combines changes from two branches, creating a new merge combining two parents.

In summary, a fast-forward merge occurs when the main branch has not diverged from the feature branch, while letting branches move forward without creating a merge commit. This is appropriate when no changes have been made to the main branch since the feature branch was created. In contrast, a 3-way merge combines changes from two branches. This is appropriate when different developers work on different branches, or when the main branch has new change commits, to record changes clearly.

- Undo, Clean, and Ignore

1. Exercise. Report the answer to step 13.

```
(base) MBP14inch2066:exercise georgew$ git show 41f61ae
commit 41f61aefc2f9188c1a5c91c1015fe29b4cfb6b2b
Author: gwang154 <gwang613@uchicago.edu>
Date:   Sat Oct 26 14:29:29 2024 -0500

    Add credentials to repository

diff --git a/credentials.txt b/credentials.txt
new file mode 100644
index 0000000..8995708
--- /dev/null
+++ b/credentials.txt
@@ -0,0 +1 @@
+supersecretpassword
(base) MBP14inch2066:exercise georgew$
```

2. Exercise. Look up `git clean` since we haven't seen this before. For context, this example is about cleaning up compiled C code, but the same set of issues apply to random files generated by knitting a document or by compiling in Python. Report the terminal output from step 7.

```
(base) MBP14inch2066:exercise georgew$ git clean -f -d
Removing README.txt~
Removing obj/
Removing src/myapp.c~
Removing src/oldfile.c~
```

3. Exercise. Report the answer to 15 ("What does git status say?")

```
(base) MBP14inch2066:exercise georgew$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    deleted:    file1.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:   .gitignore
```

Using !file3.txt in .gitignore track specific files, such as important files, even when the general rule ignores all files. Even though .gitignore now includes !file3.txt to make an exception, the file will not automatically be tracked unless explicitly added.