Git tutorial

1. To create a new repository in an **existing** directory, type the following command.

```
git init
```

2. To create a new repository in a new directory (with name `temp`, for example), type the following command. If you used this method, then don't forget to enter the directory before executing the next few commands. You can enter the directory by typing `cd temp`.

```
git init temp
```

3. To know the current status of the repository, type the following command.

```
git status
```

4. Suppose you have created a file called `question1.c` in your repository. Running `git status` should show you this output.
`bash $ git status On branch main No commits yet Untracked files: (use "git add <file>..." to include in what will be committed) question1.c nothing added to commit but untracked files present (use "git add" to track)`
Note that `question1.c` is listed under "Untracked files". This means it has not been committed to the repository.

5. Suppose the `question1.c` has the following content. We will now **commit** it to the repository.

```
#include <stdio.h>

int main() {
  return 0;
}
```

6. Run the following commands to commit `question1.c` to the repository. The string after the `-m` in the second command can be anything. It is convention to use "Initial commit" for the first commit into a git repository.

```
git add question1.c
git commit -m"Initial commit"
```

The output of the second command will look like the following.

```
$ git commit -m"Initial commit"
[main (root-commit) 1027eda] Initial commit
 1 file changed, 5 insertions(+)
 create mode 100644 question1.c
```

The 5 insertions refers to the number of lines in `question1.c`.

7. Running `git log` will show you the commit history.

```
$ git log
commit 1027eda816f8b5084b5bbe78bbe173d8c8acdd65 (HEAD -> main)
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 14:57:17 2023 +0530

    Initial commit
```

8. Now suppose we modify the content to `question1.c` to add a `printf` statement.

```
#include <stdio.h>

int main() {
  printf("Hello world!\n");
  return 0;
}
```

9. Running `git status` will tell you that the repository has been modified.

```
$ git status
On branch main
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:   question1.c

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

10. You can look at the changes by running `git diff`. The `+` sign in front of the `printf` indicates that the line has been added. Deleted lines will be prefixed with a `-` sign.

```
$ git diff
diff --git a/question1.c b/question1.c
index 1394ce8..7791fbb 100644
--- a/question1.c
+++ b/question1.c
```

```
@@ -1,5 +1,6 @@
 #include <stdio.h>

 int main() {
+  printf("Hello world!\n");
   return 0;
 }
```

11. Commit the changes to `question1.c` by running the following commands.

```
$ git add question1.c
$ git commit "Added printf"
```

12. The output of the commit will look like the following.

```
$ git commit -m"Added printf"
[main 088899b] Added printf
 1 file changed, 1 insertion(+)
```

13. In the previous step, since `question1.c` is already committed in the repository, you can combined the `add` and `commit` operations with `git commit -am"Added printf"`. Note the extra `a` after the hyphen.

14. Running `git log` will show you the updated commit history which has two commits.

```
$  git log
commit 088899bdac01714ba13d0e28cf350635f2694a4a (HEAD -> main)
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 15:04:08 2023 +0530

    Added printf

commit 1027eda816f8b5084b5bbe78bbe173d8c8acdd65
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 14:57:17 2023 +0530

    Initial commit
```

15. Suppose you want to go back to a previous state. Maybe you had a working program which is not working now. In our running example, we will go back to the first commit of `question1.c`. Note the hexadecimal string above the "Author" name in the `git log` output. It is `1027eda816f8b5084b5bbe78bbe173d8c8acdd65` in our example.

16. Run the following command to create a new **branch** with the old code. Note that we used only the first four characters in the hexadecimal string. The latest code will be in the existing branch which will be called `master` or `main` in most systems.

```
$ git checkout 1027 -b oldcode
Switched to a new branch 'oldcode'
```

17. Running `git log` will show only one commit now.

```
$ git log
commit 1027eda816f8b5084b5bbe78bbe173d8c8acdd65 (HEAD -> main)
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 14:57:17 2023 +0530

    Initial commit
```

18. Running `git branch` will show that there are two branches with names `main` and `oldcode`. You can switch between them with the `git checkout` command. The `*` next to `oldcode` indicates that we are currently in the `oldcode` branch.

```
$ git branch
  main
* oldcode
```

19. Print out the contents to `question1.c` to see that the `printf` is missing.

```
$ cat question1.c
#include <stdio.h>

int main() {
  return 0;
}
```

20. You can go back to the latest version of the program by typing `git checkout main`

```
$ git checkout main
Switched to branch 'main'
```

21. Print out the contents to `question1.c` to see that the `printf` is back.

```
$ cat question1.c
#include <stdio.h>
```

```
int main() {
  printf("Hello world!\n");
  return 0;
}
```

22. Run `git log` to see that both commits are now present.

```
$ git log
commit 088899bdac01714ba13d0e28cf350635f2694a4a (HEAD -> main)
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 15:04:08 2023 +0530

    Added printf

commit 1027eda816f8b5084b5bbe78bbe173d8c8acdd65 (oldcode)
Author: Saravanan Vijayakumaran <sarva.v@gmail.com>
Date:   Mon Mar 13 14:57:17 2023 +0530

    Initial commit
```