2. Create two additional test branches in your project, each of which is branched directly from *master*. **For each of the following cases, write the command and the corresponding output message of Git, in Markdown language in a `readme.md` file inside your `homework/1/` folder in your master branch**. If you don't have this folder in your project, then create the folder and then place your initial `readme.md` file in this folder with your signature as the content of the file. Then stage and commit this file together with homework-1 folder to your local repository. Throughout the rest of this homework, you will fill this `readme.md` file with your answers.

(A) Create two branches, **both from your *master* branch**, with names *test1* and *test2*.

```bash
git branch test1
git branch test2
```

(B) Now checkout the *test1* branch and create a new text file named `test.txt` in the `homework/1/` directory of this branch.

```bash
git checkout test1
vim test.txt
```

(C) Inside `test.txt` in *test1* branch write this message: *This is some example text for branch test1*, and save it.

```bash
this is some example text for branch test1
 :wq
```

(D) Now stage and commit `test.txt` file to branch *test1*.

```bash
git add --all
git commit
```

(E) Checkout the branch *test2*. Do you still see `test.txt` that you just created in your `homework/1/` directory? You can search for it by the *bash* command `ls`. Explain why you see/don't see the file in your working directory anymore.

it was made in a different parallel branch, if it were in the master branch it would be visible.

(F) Create a new text file named `test.txt` in the `homework/1/` directory of this branch as well, and add *This is some example text for branch test2* to its content.

```bash
vim test.txt
This is some example text for branch test2
```

(G) Now try to checkout *test1*. What error/warning message do you get? Fix the source of error and then checkout *test1* branch.

```
error: The following untracked working tree files would be overwritten by checkout:
        Homework/1/test.txt
```

Please move or remove them before you switch branches.

```
Aborting
```

```bash
git add --all
git commit -m"commiting test2 branch changes"
```

(H) Now merge the content of *test1* with *master* branch. (Hint: Note from which branch you doing this merge!)

```bash
git checkout master
git merge test1
```

(I) Now what do you see as the content of *master* branch? (Hint: Use `ls` *bash*command, to list the files in the working directory.)

readme.md test.txt

(J) Now merge the content of *test2* with *master* branch. What error/warning message do you get? Why does this error arise?

```
Auto-merging Homework/1/test.txt
CONFLICT (add/add): Merge conflict in Homework/1/test.txt
```

```bash
git merge test2
```

both branches test1 and test2 contain files test.txt- in order to merge they must not overlap

---

(K) Now chekcout *test2*. What error/warning message do you get?

```
Homework/1/test.txt: needs merge
```

error: you need to resolve your current index first

(L) Run the Git command `git status`. Why does such a conflict exist, as mentioned in `git status` output?

test.txt has not been merged.

(You have unmerged paths.

(fix conflicts and run "git commit")

(use "git merge --abort" to abort the merge)

Unmerged paths:

(use "git add <file>..." to mark resolution)

both added: test.txt)

(M) At this stage, you have two options: Either 1. stage and commit the combined conlifting `test.txt` file to Git repository (but this is not recommended), or, 2. open the file `test.txt` using *vim* editor on the command line and reslve the conflict by editing the content of the file to only this sentence: *. Then save and quit *vim*.

```bash
vim test.txt
```

deleted the second sentence and git notes that were inserted

(N) Now, run `git status`, then stage and commit your conflict-resolved file. Then checkout *test2* branch.

```bash
git status
git add --all
git commit
git checkout test2
```

(O) Now, try deleting branch *test1*, while on branch *test2*. What error/warning message do you get?

```
error: The branch 'test1' is not fully merged.
```

(P) Now, switch back to *master* branch. Now, try deleting branch *test1*, while on *master* branch. What message do you get from Git? List all the existing branches using `git branch` command.

```
Deleted branch test1 (was 7fd7916).
* master
  test2
```

(Q) Why is there such a difference in Git messages between when you tried deleting *test1* branch from *test2* branch, and when you tried deleting *test1* branch from *master* branch?

test1 and test2 branches are not connected. master contains both branches. master has been merged with the test branches while the test branches have not been merged together

---

(R) Now checkout *test2* branch. While on *test2*, try to delete branch *test2*. What error/message do you get?

```bash
error: Cannot delete branch 'test2' checked out at '/Users/Brent/Documents/git/ICP2017F'
```

(S) Switch back to *master* and delete *test2* branch. List all your project branches by the appropriate Git command.

```bash
git branch –d test2
```

```
git branch
```

* master

(T) Stage and commit all the changes (including the file `test.txt`) to your project's *master* branch. Now push it all to the remote repository by Wednesday Feb 15 2017, 9:00 a.m. CDT.

```bash
git add –all
git commit
git push ––all
```