NAUT B05 Assessment Scenario

NAME: REUBEN THOMAS
PS NO: 10685230
BATCH: 5

Note: Paste the final screenshot for every question once completed

Lab time: 8hrs

GIT

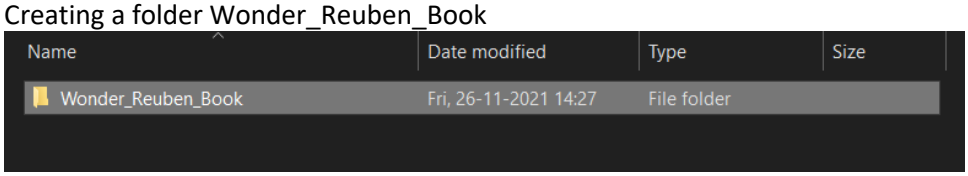
Demonstrate the following – <Must follow the sequence>

1. You are working on a creating a masterpiece 'Wonder <Your Name> book' of your own and keeping track of changes you make using git for local repository and GitHub for remote repo. (keep track of text files.). Remember – keep remote repository updated continuously.
2. Create a copy of your Remote repo. On your base machine.
3. While working on your project – demonstrate git lifecycle with explanations of steps you are taking.
4. You want to try a new 'Marvel theme' to be added in your book and simultaneously keep working on your current story. How will you handle the situation? And involve the work on new theme ones is it completed with the main book. - Demonstrate.
5. Check and display the GitHub remote repositories to which the local repository is linked to in git bash.
   <Note – Keep updating your remote repository>
6. Get all the changes in the remote repository in the copy of it which you created some time back.
7. Explain purpose of git tag and demonstrate it.
8. Demonstrate the purpose of git revert, git stash, git rebase, squash.

Outputs:

1.

Creating a folder Wonder_Reuben_Book



Initially the folder is empty. I created the chapter1.txt file and made the changes in the file.

```
MINGW64:/d/LTI Work Related/Scenario/Wonder_Reuben_Book

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ ls

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ touch chapter1.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ nano chapter1.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ ls -la
total 1
drwxr-xr-x 1 Reuben 197609  0 Nov 26 14:30 ./
drwxr-xr-x 1 Reuben 197609  0 Nov 26 14:27 ../
-rw-r--r-- 1 Reuben 197609 73 Nov 26 14:31 chapter1.txt
```

Now we want git to track all the files we create locally. Hence, we first need to initialize git for it to track any changes made in the local system.

We use the git init command to initialize it.

```
MINGW64:/d/LTI Work Related/Scenario/Wonder_Reuben_Book

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ git init
Initialized empty Git repository in D:/LTI Work Related/Scenario/Wonder_Reuben_Book/.gi
t/

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$
```

We can see that now the .git file is created, meaning now we are ready to use git to track our changes.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls -la
total 5
drwxr-xr-x 1 Reuben 197609  0 Nov 26 14:31 ./
drwxr-xr-x 1 Reuben 197609  0 Nov 26 14:27 ../
drwxr-xr-x 1 Reuben 197609  0 Nov 26 14:31 .git/
-rw-r--r-- 1 Reuben 197609 73 Nov 26 14:31 chapter1.txt
```

Git status shows the files that are modified and are yet to be put to the staging area. The below screenshot shows that chapter1 is modified but not yet staged.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        chapter1.txt

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

To stage the file, we used the git add command.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git add .
warning: LF will be replaced by CRLF in chapter1.txt.
The file will have its original line endings in your working directory

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   chapter1.txt
```

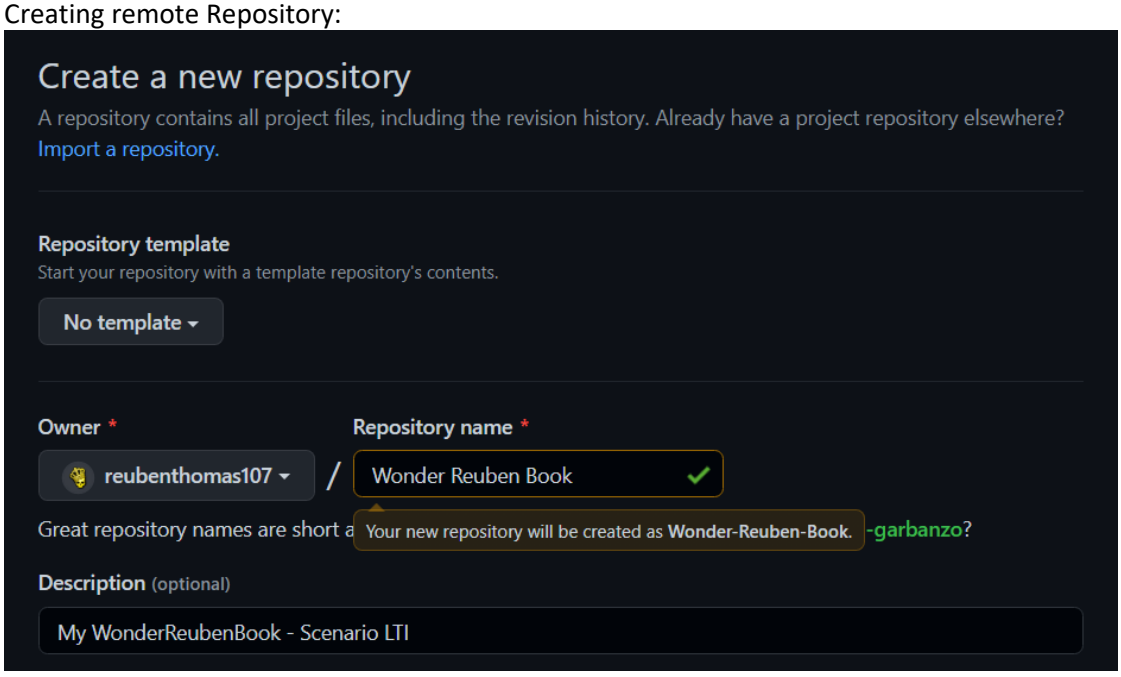
Once the changes are staged successfully, we commit the changes. Committing the changes means we are storing the changes in the local repository.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git commit -m "Chapter1 complete story"
[master (root-commit) fd93926] Chapter1 complete story
 1 file changed, 1 insertion(+)
 create mode 100644 chapter1.txt
```

To see the changes committed, we use git log. Git log shows the entire history of all the commits made.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git log
commit fd9392681f3b8d0122dacac84bd0c171cae70616 (HEAD -> master)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:36:07 2021 +0530

    Chapter1 complete story
```

Creating remote Repository:



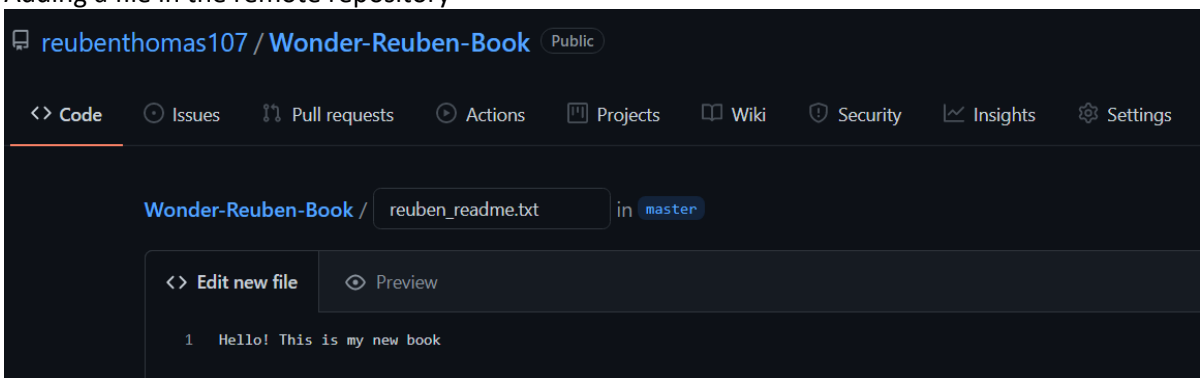
Adding the remote path to my remote repository.

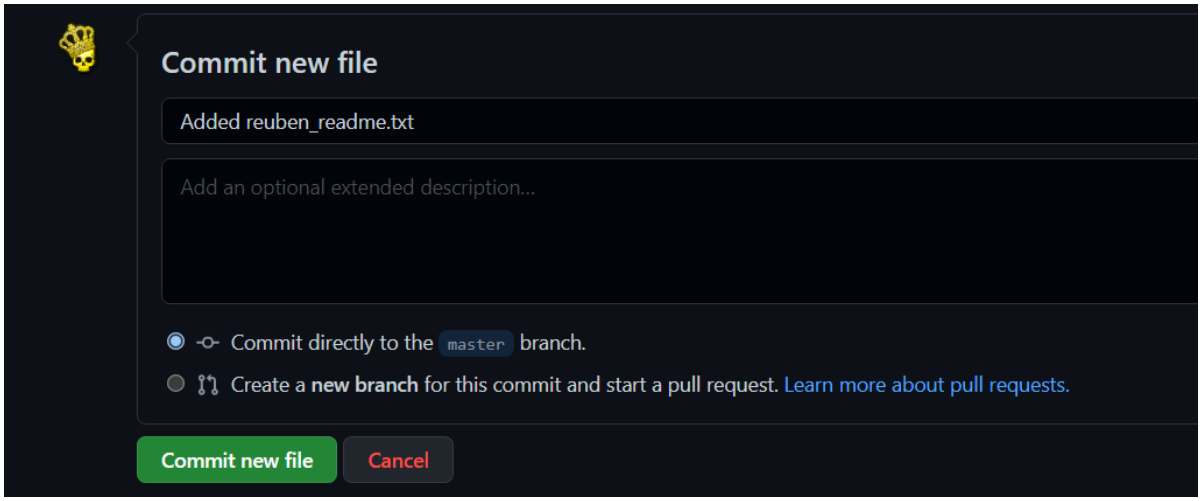
```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git remote add origin https://github.com/reubenthomas107/Wonder-Reuben-Book.git

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git branch -M master
```

Pushing the changes from the local repository to the remote repository

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Adding a file in the remote repository





Updating local repository with remote using git pull command.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls
chapter1.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git pull origin master
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 705 bytes | 3.00 KiB/s, done.
From https://github.com/reubenthomas107/Wonder-Reuben-Book
 * branch            master     -> FETCH_HEAD
   fd93926..8fdc850  master     -> origin/master
Updating fd93926..8fdc850
Fast-forward
 reuben_readme.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 reuben_readme.txt
```

The local repository is now synced with the remote repository.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls
chapter1.txt  reuben_readme.txt
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git log
commit 8fdc850d1688095c5d8ea318d9b7f502a4c78094 (HEAD -> master, origin/master)
Author: Reuben Thomas <56154927+reubenthomas107@users.noreply.github.com>
Date:   Fri Nov 26 14:42:04 2021 +0530

    Added the reuben_readme.txt

commit fd9392681f3b8d0122dacac84bd0c171cae70616
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:36:07 2021 +0530

    Chapter1 complete story
```

Adding more files and commits in the local repository

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ touch chapter2.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ nano chapter2.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git add .
warning: LF will be replaced by CRLF in chapter2.txt.
The file will have its original line endings in your working directory

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git commit -m "Chapter2 complete story"
[master 9ec5685] Chapter2 complete story
 1 file changed, 1 insertion(+)
 create mode 100644 chapter2.txt
```

Chapter 3 added and put to staging area. The files from the staging area are committed to the local repository when a commit command is executed.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        chapter3.txt

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

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git add .
warning: LF will be replaced by CRLF in chapter3.txt.
The file will have its original line endings in your working directory
```

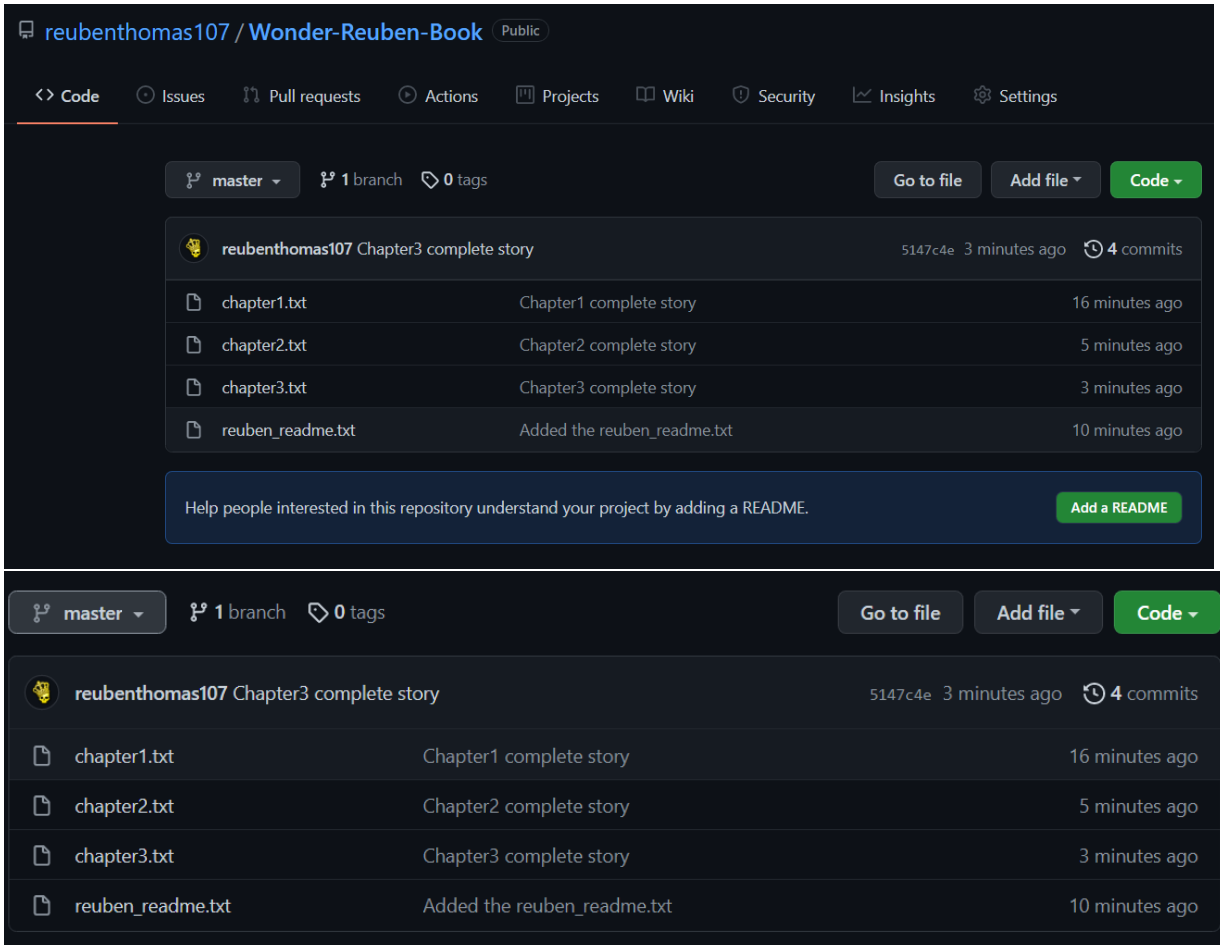
Log of all the files committed up till now.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git log
commit 5147c4e4e960e1b69cecda8fd1efefa4cfd7dda5 (HEAD -> master)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:48:46 2021 +0530

    Chapter3 complete story

commit 9ec56854e9f1c5e714614dbf97a5f3f7c668191f
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:46:14 2021 +0530

    Chapter2 complete story

commit 8fdc850d1688095c5d8ea318d9b7f502a4c78094 (origin/master)
Author: Reuben Thomas <56154927+reubenthomas107@users.noreply.github.com>
Date:   Fri Nov 26 14:42:04 2021 +0530

    Added the reuben_readme.txt

commit fd9392681f3b8d0122dacac84bd0c171cae70616
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:36:07 2021 +0530

    Chapter1 complete story
```

To update the remote repository with the local repository we use the git push command:

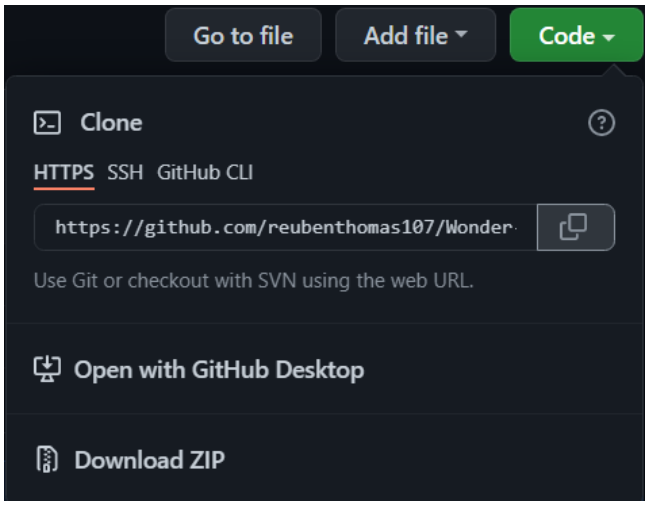
```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git push -u origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 635 bytes | 635.00 KiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
   8fdc850..5147c4e  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

We can see the changes are reflected in the GitHub dashboard.



2.

Copy of remote on base machine. This is done using the git clone command. We obtain our repository URL from here.



Git clone is done on a backup folder location.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook
$ git clone https://github.com/reubenthomas107/Wonder-Reuben-Book.git
Cloning into 'Wonder-Reuben-Book'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 12 (delta 1), reused 9 (delta 1), pack-reused 0
Receiving objects: 100% (12/12), done.
Resolving deltas: 100% (1/1), done.
```

All the files and commit history is reflected in this.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook
$ ls
Wonder-Reuben-Book/

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook
$ cd Wonder-Reuben-Book/

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reub
en-Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  reuben_readme.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reub
en-Book (master)
$ git log
commit 5147c4e4e960e1b69cecda8fd1efefa4cfd7dda5 (HEAD -> master, origin/master, origin/HEAD)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:48:46 2021 +0530

    Chapter3 complete story

commit 9ec56854e9f1c5e714614dbf97a5f3f7c668191f
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:46:14 2021 +0530

    Chapter2 complete story
```

3. (Covered in the doc)

4. Adding new “Marvel Theme”

Creating a new branch from chapter3, all the files from that commit are present

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git branch marveltheme

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git branch
  marveltheme
* master

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git checkout marveltheme
Switched to branch 'marveltheme'
```

Files modified with new marvel theme:

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git branch
* marveltheme
  master

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git status
On branch marveltheme
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:   chapter1.txt
        modified:   chapter2.txt
        modified:   chapter3.txt

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

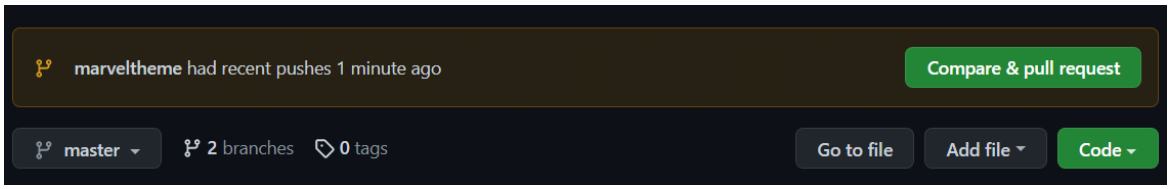
Commit the changes in the new branch.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git commit -m "Marvel_Theme Added for Chapter 1,2,3"
[marveltheme cebe717] Marvel_Theme Added for Chapter 1,2,3
 3 files changed, 3 insertions(+)

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git log
commit cebe71728a3d35ed7716e047a0f3b6e79450e23a (HEAD -> marveltheme)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:24:25 2021 +0530

    Marvel_Theme Added for Chapter 1,2,3

commit 5147c4e4e960e1b69cecda8fd1efefa4cfd7dda5 (origin/master, master)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:48:46 2021 +0530

    Chapter3 complete story
```

Updating the remote repository both master and the branch

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git push -u origin marveltheme
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 588 bytes | 588.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'marveltheme' on GitHub by visiting:
remote:      https://github.com/reubenthomas107/Wonder-Reuben-Book/pull/new/marveltheme
remote:
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
 * [new branch]      marveltheme -> marveltheme
Branch 'marveltheme' set up to track remote branch 'marveltheme' from 'origin'.
```

Reflected changes in the github website.



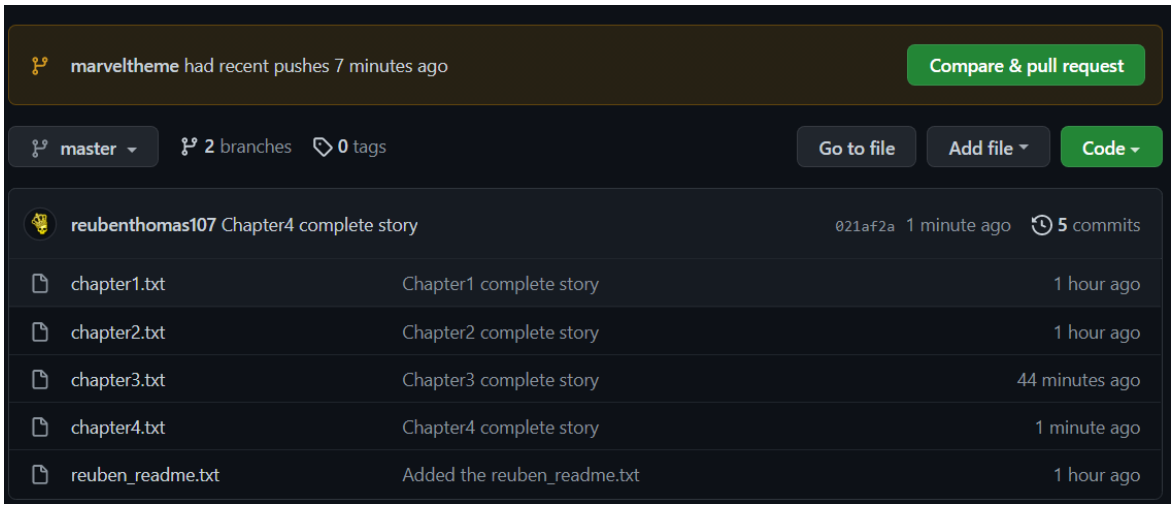
Switching back to master using git switch command.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (marveltheme)
$ git switch master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

Added another chapter - chapter4

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        chapter4.txt

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

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git add .
warning: LF will be replaced by CRLF in chapter4.txt.
The file will have its original line endings in your working directory
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git commit -m "Chapter4 complete story"
[master 021af2a] Chapter4 complete story
 1 file changed, 1 insertion(+)
 create mode 100644 chapter4.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 324 bytes | 324.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
   5147c4e..021af2a  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```



We pushed all the changes to the remote repository

Now, we Merge the branches of the master and the marvel theme to apply the new theme. Using the branches, we were able to **simultaneously work** on other parts and merge the changes to the original from the branch.

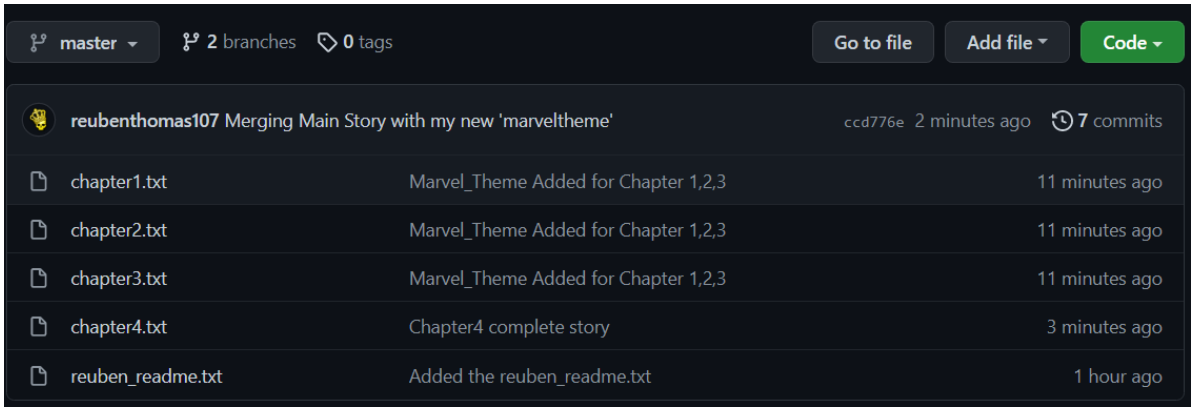
```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git branch
  marveltheme
* master

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git merge marveltheme
Merge made by the 'ort' strategy.
 chapter1.txt | 1 +
 chapter2.txt | 1 +
 chapter3.txt | 1 +
 3 files changed, 3 insertions(+)
```

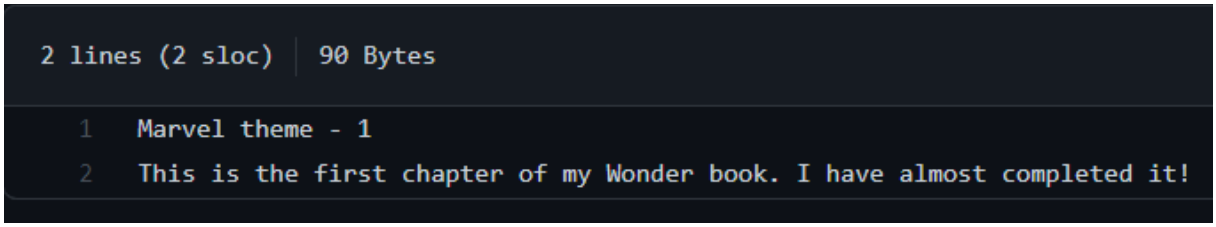
All the files merged successfully. We can see the changes

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  reuben_readme.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ cat chapter1.txt
Marvel theme - 1
This is the first chapter of my Wonder book. I have almost completed it!
```

Update the remote repository with the local repository.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git push -u origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 313 bytes | 313.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
   021af2a..ccd776e  master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```





5. Checking to see if the repository is up to date.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git push
Everything up-to-date

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git log
commit ccd776e82867ab65755a3400ffa7b630f0c3a3d2 (HEAD -> master, origin/master)
Merge: 021af2a cebe717
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:33:31 2021 +0530

    Merging Main Story with my new 'marveltheme'

commit 021af2a8e5abc7d83f8c9561ca24a3aadc433df1
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:31:55 2021 +0530

    Chapter4 complete story
```

git remote lets us know which repository we are **connected/linked** to in git bash.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git remote -v
origin  https://github.com/reubenthomas107/Wonder-Reuben-Book.git (fetch)
origin  https://github.com/reubenthomas107/Wonder-Reuben-Book.git (push)
```

6.

The Clone folder before updating the local repository

```
MINGW64:/d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reuben-Book

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reuben-Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  reuben_readme.txt
```

Before working on the clone, we do a git pull to update it from the remote repository.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reuben-Book (master)
$ git pull
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 10 (delta 3), reused 9 (delta 2), pack-reused 0
Unpacking objects: 100% (10/10), 1.04 KiB | 0 bytes/s, done.
From https://github.com/reubenthomas107/Wonder-Reuben-Book
   5147c4e..ccd776e  master      -> origin/master
 * [new branch]      marveltheme -> origin/marveltheme
Updating 5147c4e..ccd776e
Fast-forward
 chapter1.txt | 1 +
 chapter2.txt | 1 +
 chapter3.txt | 1 +
 chapter4.txt | 1 +
 4 files changed, 4 insertions(+)
 create mode 100644 chapter4.txt
```

All previous changes have now been successfully updated in our local folder.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reuben-Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  reuben_readme.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/BackupWonderReubenBook/Wonder-Reuben-Book (master)
$ cat chapter2.txt
Marvel theme - 2
This is the second chapter of my book. Hope you like it!
```

7. Using git tag

Tags are references that point to specific points in Git history. Tagging captures a point in history. A tag is just like a branch that does not change which means that once it is created there are no commits made on it.

We create a branch, ch2release

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git checkout 9ec56854e9f1c5e714614dbf97a5f3f7c668191f
Note: switching to '9ec56854e9f1c5e714614dbf97a5f3f7c668191f'.

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 9ec5685 Chapter2 complete story

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book ((9ec5685...))
$ git branch ch2release
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git log
commit 9ec56854e9f1c5e714614dbf97a5f3f7c668191f (HEAD -> ch2release)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:46:14 2021 +0530

    Chapter2 complete story

commit 8fdc850d1688095c5d8ea318d9b7f502a4c78094
Author: Reuben Thomas <56154927+reubenthomas107@users.noreply.github.com>
Date:   Fri Nov 26 14:42:04 2021 +0530

    Added the reuben_readme.txt

commit fd9392681f3b8d0122dacac84bd0c171cae70616
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:36:07 2021 +0530

    Chapter1 complete story
```

This contains only original chapter 1 and 2, we use tag to this branch to preserve it as a release. Git tag <name> is used to tag it.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git tag release_chp_1_and_2

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git tag
release_chp_1_and_2

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git show
commit 6b2d76ed3e7b15435b70f67a7e4eff6b71590a3b (HEAD -> ch2release, tag: release_chp_1_and_2)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:57:46 2021 +0530

    Information file added
```

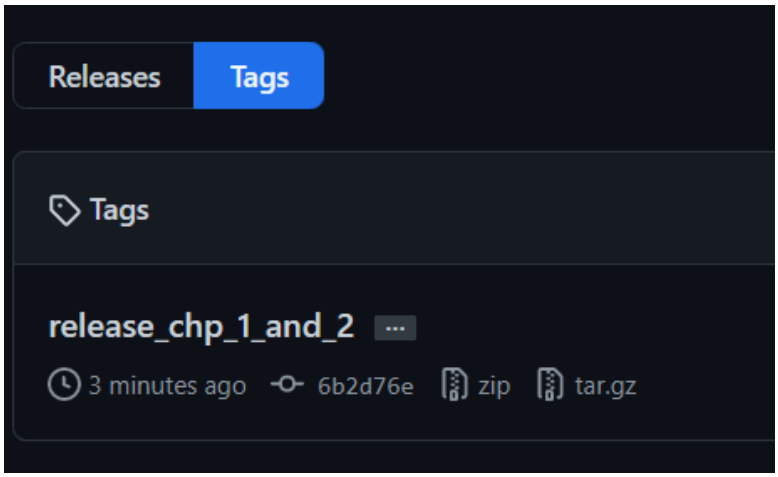
We now see the tag added to the log.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git show release_chp_1_and_2
commit 6b2d76ed3e7b15435b70f67a7e4eff6b71590a3b (HEAD -> ch2release, tag: release_chp_1_and_2)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:57:46 2021 +0530

    Information file added

diff --git a/info.txt b/info.txt
new file mode 100644
index 0000000..987da8e
--- /dev/null
+++ b/info.txt
@@ -0,0 +1 @@
+This is the original theme release for chapter 1 and 2
```

We now push the changes to the remote repository.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch2release)
$ git push origin release_chp_1_and_2
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes | 322.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/reubenthomas107/Wonder-Reuben-Book.git
 * [new tag]         release_chp_1_and_2 -> release_chp_1_and_2
```

Tag visible on the GitHub dashboard



8.

git revert

Revert is used to invert changes made by a commit but it does this invert by writing a commit. This preserves the history for future reference.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  reuben_readme.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ touch chapter5.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ nano chapter5.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git status
On branch ch4test
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        chapter5.txt

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

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git add .
warning: LF will be replaced by CRLF in chapter5.txt.
The file will have its original line endings in your working directory

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git commit -m "Chapter5 incomplete story"
[ch4test 929065d] Chapter5 incomplete story
 1 file changed, 1 insertion(+)
 create mode 100644 chapter5.txt
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter5 incomplete story
```

I want to revert back the previous chapter5 commit. Since I don't want that commit(chapter5.txt) due to some issue.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit 45a5f63ba259eaf3ea74e516fbd2fd40abf7e316 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:08:24 2021 +0530

    Chapter6 added

commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter5 incomplete story

commit 021af2a8e5abc7d83f8c9561ca24a3aadc433df1
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:31:55 2021 +0530

    Chapter4 complete story
```

Git revert <hash> at the point you want to revert.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git revert 929065ddf8fd29a8cd3bf69bb323090e575f07e0
[ch4test ed31afb] Revert "Chapter5 incomplete story"
 1 file changed, 1 deletion(-)
 delete mode 100644 chapter5.txt
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit ed31afb221bff337815cb87d46ad421ecf00f273 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:09:19 2021 +0530

    Revert "Chapter5 incomplete story"

    This reverts commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0.

commit 45a5f63ba259eaf3ea74e516fbd2fd40abf7e316
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:08:24 2021 +0530

    Chapter6 added

commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter5 incomplete story
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter6.txt  reuben_readme.txt
```

The chapter5.txt was reverted due to the revert in the commit that added it.

Git stash:

git stash stashes the changes you've made to your working copy so you can work on something else, and then come back and re-apply them later on with doing a commit.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter6.txt  reuben_readme.txt
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git add .
warning: LF will be replaced by CRLF in chapter5.txt.
The file will have its original line endings in your working directory

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git stash save "Chapter5 being worked on in ch4test"
Saved working directory and index state On ch4test: Chapter5 being worked on in ch4test
```

Git stash list is used to list all the available stashes

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git stash list
stash@{0}: On ch4test: Chapter5 being worked on in ch4test
```

Git stash apply does not delete the stash unlike in pop.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git stash apply stash@{0}
On branch ch4test
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   chapter5.txt
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter3.txt  chapter5.txt  reuben_readme.txt
chapter2.txt  chapter4.txt  chapter6.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ cat chapter5.txt
New chapter in my story!
```

Stashing is done before switching to another branch, this allows us to save our work without having to commit the changes.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ nano chapter7.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter7.txt  reuben_readme.txt

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git add .
warning: LF will be replaced by CRLF in chapter7.txt.
The file will have its original line endings in your working directory

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   chapter7.txt
```

We created a stash and all our uncommitted work is saved in that stash.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git stash save "Chapter7 being worked on in master"
Saved working directory and index state On master: Chapter7 being worked on in master

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git stash list
stash@{0}: On master: Chapter7 being worked on in master
stash@{1}: On ch4test: Chapter5 being worked on in ch4test
```

Git stash pop remove it from the list when you access it.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ git stash pop stash@{0}
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   chapter7.txt

Dropped stash@{0} (e84739227c6a4d714544a8c8d4169ef86cdb605e)

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (master)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter7.txt  reuben_readme.txt
```

On pop, we retrieved the chapter7 file which was stashed.

Git squash:

Top 3 commits are too much history, we can reduce the history by combining them into a single commit. This is done using squash.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit ed31afb221bff337815cb87d46ad421ecf00f273 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:09:19 2021 +0530

    Revert "Chapter5 incomplete story"

    This reverts commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0.

commit 45a5f63ba259eaf3ea74e516fbd2fd40abf7e316
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:08:24 2021 +0530

    Chapter6 added

commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter5 incomplete story

commit 021af2a8e5abc7d83f8c9561ca24a3aadc433df1
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:31:55 2021 +0530

    Chapter4 complete story
```

```
MINGW64:/d/LTI Work Related/Scenario/Wonder_Reuben_Book
pick 929065d Chapter5 incomplete story
squash 45a5f63 Chapter6 added
squash ed31afb Revert "Chapter5 incomplete story"

# Rebase 021af2a..ed31afb onto 021af2a (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                    commit's log message, unless -C is used, in which case
#                    keep only this commit's message; -c is same as -C but
#                    opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified); use -c <commit> to reword the commit message
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
</Scenario/Wonder_Reuben_Book/.git/rebase-merge/git-rebase-todo[+] [unix] (1
```

Another window with the comments to be added was saved and the below output was generated.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git rebase -i HEAD~3
[detached HEAD 3891807] Chapter 5,6 added and reverted Chapter5
 Date: Fri Nov 26 16:05:41 2021 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 chapter6.txt
Successfully rebased and updated refs/heads/ch4test.
```

All the 3 commits were combined into a single commit

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit 389180706d5ee392e73e989f33bdd0440b634ff1 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter 5,6 added and reverted Chapter5

    This reverts commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0.

commit 021af2a8e5abc7d83f8c9561ca24a3aadc433df1
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:31:55 2021 +0530

    Chapter4 complete story

commit 5147c4e4e960e1b69cecda8fd1efefa4cfd7dda5
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 14:48:46 2021 +0530

    Chapter3 complete story
```

Git rebase:

Before:

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter6.txt  reuben_readme.txt
```

Performing rebase on the ch4test branch.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git rebase master
Successfully rebased and updated refs/heads/ch4test.
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ git log
commit 9eb346403ce70ceb421a8f224f5fc4ece9813dd5 (HEAD -> ch4test)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:05:41 2021 +0530

    Chapter 5,6 added and reverted Chapter5

    This reverts commit 929065ddf8fd29a8cd3bf69bb323090e575f07e0.

commit 20d7fc12ecc32e4d5c155a3616b05da876e08a09 (origin/master, master)
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 16:22:21 2021 +0530

    Chapter7 added

commit ccd776e82867ab65755a3400ffa7b630f0c3a3d2
Merge: 021af2a cebe717
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:33:31 2021 +0530

    Merging Main Story with my new 'marveltheme'

commit 021af2a8e5abc7d83f8c9561ca24a3aadc433df1
Author: reuben <reubenthomas107@gmail.com>
Date:   Fri Nov 26 15:31:55 2021 +0530

    Chapter4 complete story
```

This branch has now been rebased with the master. Chapter 1,2,3,4 and 7 from master have rebased.

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book
$ cat chapter2.txt
Marvel theme - 2
This is the second chapter of my book. Hope you like it!
```

```
Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ cat chapter7.txt
Last file of my story, nothing more to come!

Reuben@LAPTOP-FIFQOML1 MINGW64 /d/LTI Work Related/Scenario/Wonder_Reuben_Book (ch4test)
$ ls
chapter1.txt  chapter2.txt  chapter3.txt  chapter4.txt  chapter6.txt  chapter7.txt  reuben_readme.txt
```

Data Structure

Q1. Write Pseudocode for the following

A. Push n Pop operation in circular queue
B. Search in a binary search tree
C. Traverse stack
D. Search and insertion in linked list
E. Deletion and insertion in doubly linked list

Q2. Write a logic or algorithm/pseudocode for Converting an infix expression to prefix and postfix expression using Stack

Q3. Write a logic or algorithm/pseudocode for checking parenthesis matching in an expression

Answers

Q.1 Write Pseudocode for the following
A. Push and Pop operation in circular queue

Push (Inserting/Enque in Circular Queue)

1. IF (front==(rear+1) % maxsize)
2. THEN, print("Queue already full, Overflow")

3. IF (front == -1 && rear == -1)
4. THEN, set front=0 and rear=0
5. ELIF (front != 0 && rear == maxsize – 1)
6. THEN, set rear=0
7. ELSE
8. THEN, set rear=(rear+1) % maxsize
9. Update the value, a[rear] = x

Pop (Deleting/Deque in circular Queue)

1. IF (front == -1)
2. THEN, print("Queue is already empty, Underflow")

3. IF (front == rear)
4. THEN, set front=rear=-1
5. ELIF (front == maxsize-1 && rear != maxsize-1)
6. THEN, set front = 0
7. ELSE, set front = front + 1

B. Search operation in a binary search tree

1. Searching through the tree for the value stored in 'val'.
2. SET node *current = root
3. WHILE(current->data != val)
    a. IF (current->data > val)
    b. THEN, set current = current->left
    c. ELSE (current->data < val)
    d. THEN, set current = current->right

    e. IF (current == null)
    f. THEN, return null
4. Finally, return current

C. Traverse stack

1. To traverse a stack in array implementation, we need the value of 'top'.
2. IF top == -1
3. THEN, print("Stack is Empty")

4. START, FOR LOOP (x=top; x>=0; x--)
    a. PRINT (stack[x])

D. Search and Insertion in Linked List

Search:

1. SET node *list with the start of the list(head), pos=0, count=0 and value to search is 'val'
2. WHILE (list != NULL)
    a. IF (list->data == val)
    b. THEN, Set count=count+1, pos = pos + 1 {can print the location in the linked list}
    print("Searched value found at location {pos}")

    c. ELSE
    SET list=list->next
3. IF (count==0)
4. THEN, print ("Value not found in the linked list")

Insert:

Insert at the start:

1. SET node *newnode and the value to insert is 'val', head is the main list.
2. newnode->data = val
3. newnode->next = head
4. SET head=newnode

Insert at the end:

1. SET node *newnode and the value to insert is ‘val’, head is the main list.
2. newnode->data = val
3. If (head == NULL)
4. THEN,
   newnode->next = NULL
   head=newnode

Insert at a specific location:

1. SET node *newnode, *list, pos (location to insert) and the value to insert is ‘val’, head is the main list.
2. newnode->data = val
3. list = head
4. LOOP (x=0; x<pos;x++)
    a. IF (list != NULL)
       list=list->next
5. newnode->next = list->next
6. list->next = newnode

D. Deletion and Insertion in Doubly Linked List

Deletion

Deletion at the beginning:

1. SET node{*prev, *next, data}, head is the main list
2. IF (head == NULL)
   THEN, the list is already empty.

3. ELIF (head->next == NULL)
   THEN, SET head=null
   del(head)

4. ELSE
   node=head
   head = head->next
   head->prev = NULL
   del(node)

Deletion at the end:

1. SET node{*prev, *next, data}, head is the main list
2. IF (head == NULL)
   THEN, the list is already empty.

3. ELIF (head->next == NULL)
   THEN, SET head=null

del(head)

4. ELSE
   node = head
   IF(node->next !=NULL)
   THEN, node = node->next
   SET node->prev->next = NULL
   del(node)

Deletion of a specific value:

1. SET node{*prev, *next, data}, list{*prev,*next, data}, head is the main list and value to be deleted is 'val'
2. node = head
3. WHILE (node->data != val)
   node = node->next

4. list = node->next
   node->next = list->next
   list->next->prev=node
   del(list)

Insertion:
Insertion at the beginning:

1. SET node{*prev, *next, data}, head is the main list and value to be inserted is 'val'
2. IF (head==NULL)
   THEN, node->next=NULL
   node->prev=NULL
   node->data=val
   head=node

3. ELSE
   node->next=head
   node->prev=NULL
   node->data=val
   head->prev=node
   head=node

Insertion at the end:

1. SET node{*prev, *next, data}, *list, head is the main list and value to be inserted is 'val'
2. IF (head==NULL)
   THEN, node->next=NULL
   node->prev=NULL

node->data=val
head=node

3. ELSE,
   WHILE (list->next != NULL)
   list=list->next

   List->next = node
   node->prev = list
   node->data = val
   node->next = NULL

Insertion at a specific location:

1. SET node{*prev, *next, data}, *list, head is the main list, pos (location to insert) and value to be inserted is 'val'
2. LOOP FOR(x=0;x<pos;x++)
   IF (list != NULL)
   list=list->next
3. node->data = val
   node->next = list->next
   node->prev = list
   list->next = node
   list->next->prev = node

Q2. Write a logic or algorithm/pseudocode for Converting an infix expression to prefix and postfix expression using Stack

Infix to Postfix

1. Creating the precedence of the different operators, written in the precedence function
   IF (ch == '^'); return value 3
   ELIF (ch == '/' or ch == '*'); return value 2
   ELIF (ch == '+' or ch == '-'); return value 1
   ELSE return the value –1

2. Stack 'stack' will store the operators, the expression is 'exp', 'out' is the output string,top=0
3. FOR LOOP (x=0; x<exp.length; x++)
   a. SET c=exp[x]
   b. IF ((c >= 'a' and c<='z') or (c>='A' and c<='Z') or (c>='0' and c<='9'))
      THEN, out = out+c

   c. ELIF (c=='(')
      stack.push('(')

top=top+1;

d. ELIF (c==')'
   WHILE(stack[top] != '('
      i. out = out + stack[top]
      ii. stack.pop();
      iii. top=top-1

   stack.pop()

e. ELSE
   WHILE ((stack.length !=0 and prec(exp[x]) <= prec(stack[top]))
      i. out=out+stack[top]
      ii. stack.pop()
      iii. top=top-1

stack.push(c)

f. Now, pop all the remaining elements from the stack
   WHILE(stack.length !=0)
      i. out=out+stack[top]
      ii. stack.pop()
      iii. top=top-1

g. PRINT out

Infix to Prefix:

1. Creating the precedence of the different operators, written in the precedence function
   IF (ch == '^'); return value 3
   ELIF (ch == '/' or ch == '*'); return value 2
   ELIF (ch == '+' or ch == '-'); return value 1
   ELSE return the value –1

2. Stack 'stack' will store the operators, the expression is 'exp', 'out' is the output string,top=0
3. Reverse the exp, exp = rev(exp)
4. FOR LOOP (x=0; x<exp.length; x++)
   a. SET c=exp[x]
   b. IF ((c >= 'a' and c<='z') or (c>='A' and c<='Z') or (c>='0' and c<='9'))
      THEN, out = out+c

   c. ELIF (c=='('
      stack.push('('
      top=top+1;

   d. ELIF (c==')'
      WHILE(stack[top] != '('
         i. out = out + stack[top]
         ii. stack.pop();

iii. top=top-1

e. ELSE

IF ((prec(exp[x]) > prec(stack[top]))

i. stack.push(c)
top=top+1;

ELIF (exp[x]==prec(stack[top]))

i. stack.push(c)
top=top+1;

ELIF(exp[x]<prec(stack[top]))

i. out = out + stack[top]
stack.pop();
top=top-1

5. Now, pop all the remaining elements from the stack
WHILE(stack.length !=0)
   i. out=out+stack[top]
   ii. stack.pop()
   iii. top=top-1

6. prefix = rev(out)
PRINT prefix

Q3. Write a logic or algorithm/pseudocode for checking parenthesis matching in an expression

1. The expression to check is 'exp', stack is empty ie stack=[],check=TRUE
2. LOOP FOR(x=0;x<s.length;s++)
   a. c=exp[x]
   IF c == '('
   stack.push(c)

   b. ELIF (c==')')
      i. IF (stack.peek() == '(')
      stack.pop()
      ii. ELSE // (If stack is empty)
      check=FALSE
3. IF check=TRUE
   THEN, Print("Parenthesis matching")
4. ELSE
   THEN, print("Parenthesis do not match")

****