[CS200]-STT II **Fall 2020-21**

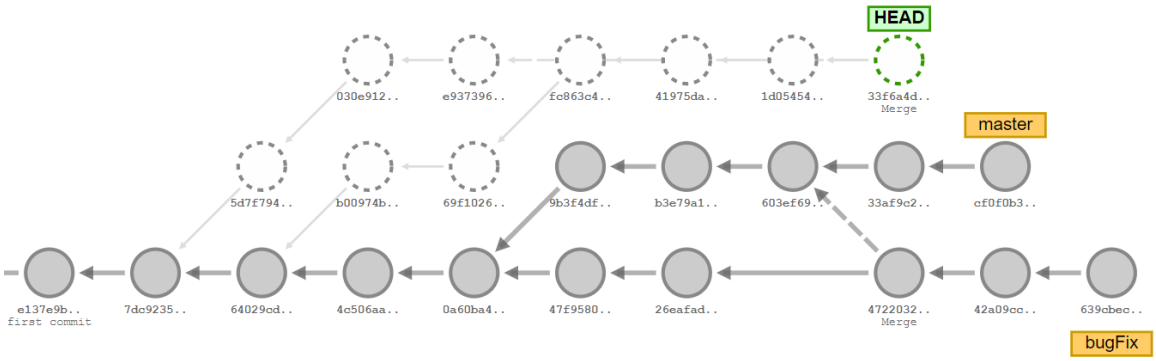
Assignment: III

Name: Santaz Sahithi RollNo: 11941140 Email: bandelas@iitbhilai.ac.in

Collaborators Names: Aayush Deshmukh(11940010), Shubham Gupta(11941140)

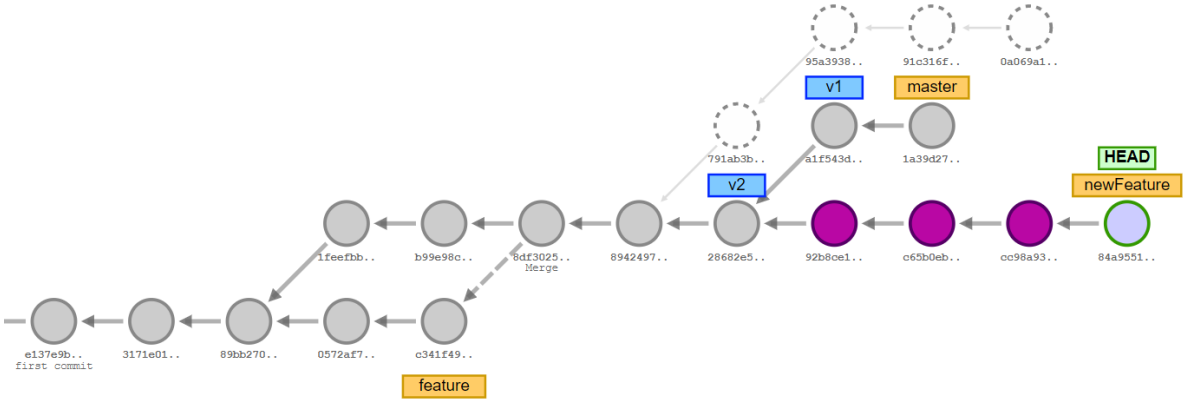
1.1 Solution of 1a

Here are the list of commands used to generate the 1a Git-Graph
-> git commit
-> git commit
-> git commit
-> git commit
-> git branch bugFix
-> git checkout bugFix
-> git commit
-> git commit
-> git checkout master
-> git commit
-> git commit
-> git commit
-> git checkout bugFix
-> git merge master
-> git commit
-> git commit
-> git checkout master
-> git commit
-> git commit
-> git checkout bugFix
-> git checkout head~8
-> git commit
-> git commit
-> git commit
-> git checkout bugFix
-> git checkout head~7
-> git commit
-> git commit
-> git checkout e937396
-> git rebase 69f1026
-> git merge e937396



1.2 Solution of 1b

Here are the list of commands used to generate the 1b Git-Graph
-> git commit
-> git commit
-> git branch checkout feature
-> git checkout feature
-> git commit
-> git commit
-> git checkout master
-> git commit
-> git commit
-> git merge feature
-> git commit
-> git commit
-> git checkout head∼1
-> git branch NewFeature
-> git checkout NewFeature
-> git commit
-> git commit
-> git commit
-> git commit
-> git rebase master
-> git checkout master
-> git commit
-> git commit
-> git checkout head∼2
-> git tag v2
-> git checkout master
-> git checkout head∼1
-> git tag v1
-> git checkout NewFeature



1.3 Solution of 1c

Since, we are merging two branches in the merge commit, so on examination of the merge commit we can see that it's hash value of the shasum will point to a tree of both the branches that are merged. In the last 38 character the string will correspond to a tree that can be checked by the hash value of the type of the object. On the occasion of a normal commit, a file with it's respective content is committed. On seeing the underlying part of the type of object will give a blob with the hash value of the 38 character string corresponding to a blob(can be verified by the hash value of the object storage type)

->echo 'test content' — git hash-object -w –stdin
The output command is a 40-character checksum hash. This is the SHA-1 hash—a checksum of the content you're storing plus a header.Now you can see how Git has stored your data.

->find .git/objects -type f
If you again examine your objects directory, you can see that it contains a file for that new content. This is how Git stores the content initially—as a single file per piece of content, named with the SHA-1 checksum of the content and its header. The subdirectory is named with the first 2 characters of the SHA-1, and the filename is the remaining 38 characters.

We have just done the low-level operations to build up a Git history without using any of the front end commands. This is essentially what Git does when you run the git add and git commit commands—it stores blobs for the files that have changed, updates the index, writes out trees, and writes commit objects that reference the top-level trees and the commits that came immediately before them. These three main Git objects—the blob, the tree, and the commit—are initially stored as separate files in your .git/objects directory".