[CS200]-Software Tools and Technologies II **Fall 2020-21**

Homework: III

Team Name: Peaky Blinders Roll Numbers:11940150, 11940370, 11940380

Solution of problem 1. The answers are provided below:

a The abstract git commands that led to git graph as given in the question are as follows:

```
#Creating temporary branches to duplicate the upper part
git commit
git branch b1
git commit
git branch b2
git commit
#Making commits in temp branches to recreate the shape in upper part
git checkout b1
git commit
git commit
git commit
git checkout b2
git commit
git commit
#Recreating the lower part
git checkout master
git commit
git branch bugFix
git checkout bugFix
git commit
git commit
git checkout master
git commit
git commit
git commit
#Recreating the exact merge commit
git checkout b2
git commit
git commit
git commit
git checkout b1
git merge b2 #Merge b2 into b1
#Complete the lower part
git checkout bugFix
git merge master
git commit
git commit
```

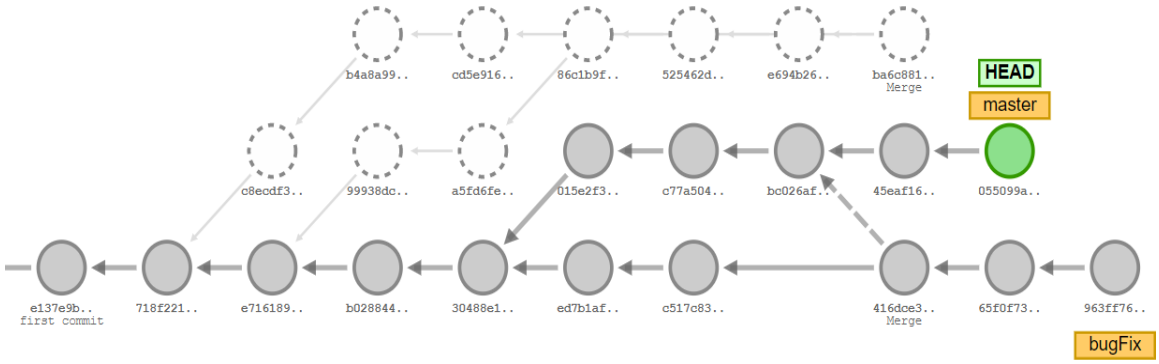

Figure 3.1: Graph obtained in q1a with above given commands

```
git checkout master
git commit
git commit
#Delete the temporary branches
git branch -d b1
git branch -d b2
```

Here you can see that we start recreating the upper part and then move to the lower part and then back again. This is done to exactly replicate the shape of the graph as it is given in the question. Also one important thing is b2 has to be merged into b1. Doing it the other way round will yield a slightly different result. The graph obtained from these commands is shown in Figure 3.1.

b The abstract git commands that led to git graph as given in the question are as follows:

```
git commit
git commit
#Create and checkout into the feature branch
git checkout -b feature
git commit
git commit
git checkout master
git commit
git commit
#Merge feature branch into master
git merge feature
git commit
#create the newFeature branch
git branch newFeature
git commit
#Make 4 commits in the newFeature branch
```

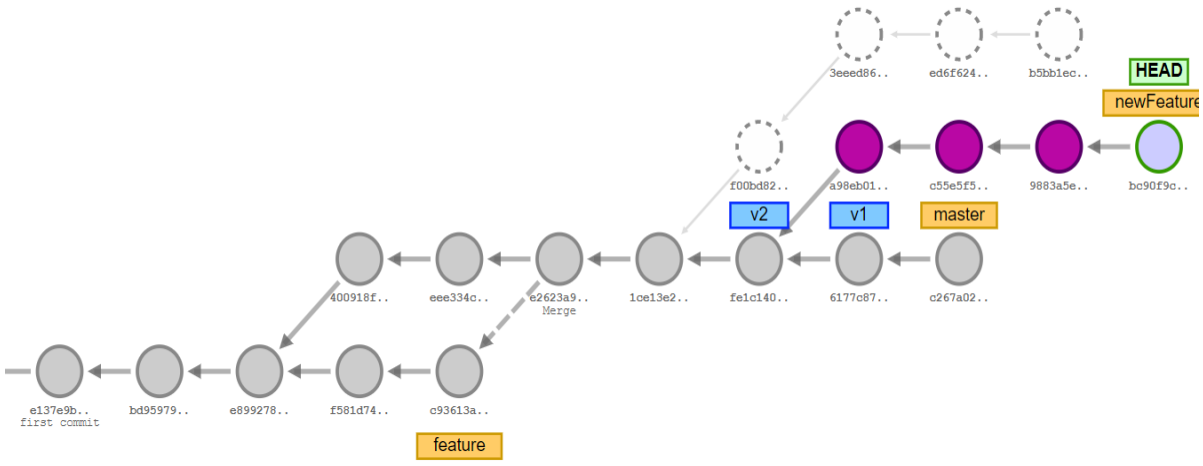

Figure 3.2: Graph obtained in q1b with above given commands

```
git checkout newFeature
git commit
git commit
git commit
git commit
git checkout master
#Tag the two commits as shown in the figure in the question
git tag v2
git commit
git tag v1
git commit
#Rebase the 4 commits done in the newFeature branch to the commit tagged v2
git checkout newFeature
git rebase v2
```

Remember if rebase is not done using the interactive option it makes copies of all the commits in current branch in same order and attaches them to the commit mentioned in the command itself(Here it was commit tagged v2). The graph obtained from these commands is shown in Figure 3.2.

c The merge commit is a special commit object because it generally has more than one parent commits as an effect of merging branches. From the Git Objects documentation we see that to calulate the shasum of an object the following steps are done: First a header is created of the format:

```
"[object-type] #{[object-content].bytesize}\0"
```

This header is concatenated with content of the object to generate a combined string. This combined string is given as a message to be encoded to the SHA-1 hash algorithm and thus the 40 character hash is generated. The contents of merge commit object are: The hash-values of parent commits, the hash-values of the trees that it

points to, the author, the committer, the timestamp and the commit message. This stored in a specific format within the commit object. All this is concatenated with the header as explained above to get the combined string and subsequently the shasum.

A small example you can try to verify this is as follows: Make a workspace folder and initialize git in it. Now make two branches and make some commits in them and finally merge them. Now making sure that HEAD points to the new merge commit created execute the following command:

```
(printf "commit %s\0" $(git cat-file commit HEAD | wc -c); git cat-file
commit HEAD) | sha1sum
```

It will give you the hash value of this merge commit. You can verify that this indeed is the commit by checking in the git objects folder and also printing the contents of this commit by using the function

```
git cat-file -p <hash-value generated by above command>
```