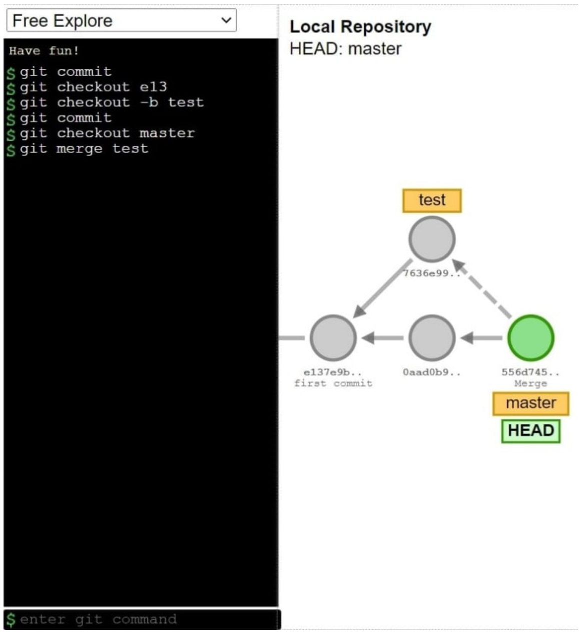
[CS200]- Software Tool and Technologies Lab-II **23-10-2020**

Homework: 3

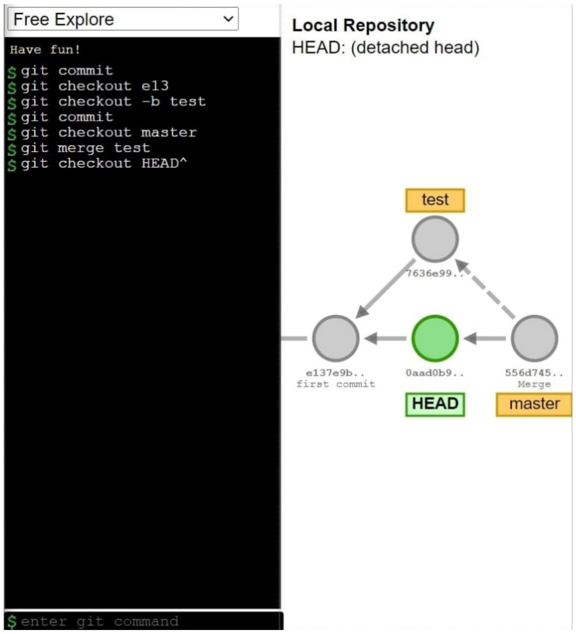
Team Name: Peaky Blinders Roll Numbers:11940150, 11940370, 11940380

Solution of Problem 2:

Git graph after operating- **git checkout HEAD^** command on both the cases are demonstrated below:-
(Figure-1.1 and 1.2)



```
Have fun!
$ git commit
$ git checkout e13
$ git checkout -b test
$ git commit
$ git checkout master
$ git merge test
$ enter git command
```

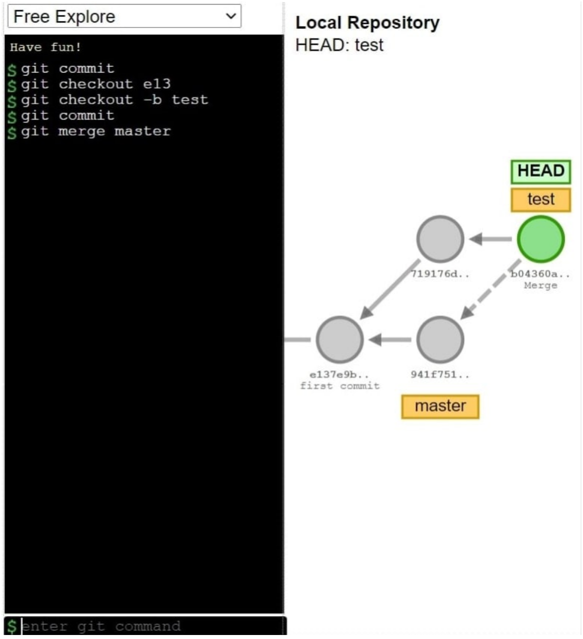


```
Have fun!
$ git commit
$ git checkout e13
$ git checkout -b test
$ git commit
$ git checkout master
$ git merge test
$ git checkout HEAD^
$ enter git command
```

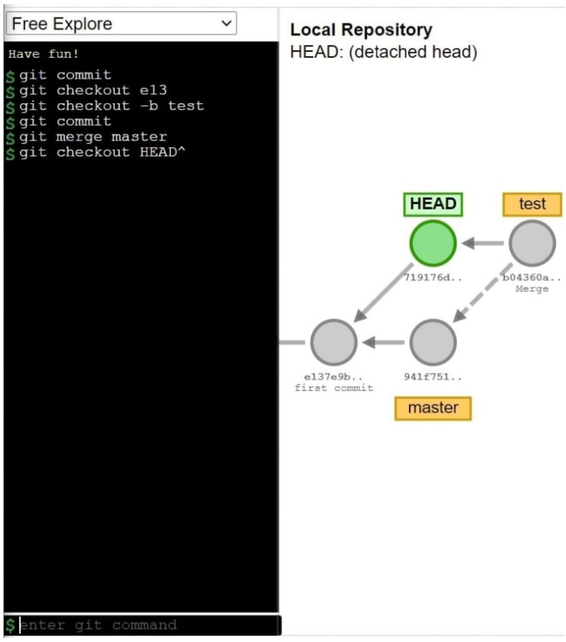
Figure 1.1: Git graph before git checkout HEAD^ Git graph after git checkout HEAD^

In the 1st case, on branch Master we execute the command:- **git merge test**
Then we combined 2 branches together so our merge commit has 2 parents. So, on running the code:-
. **git checkout HEAD^**
We would find that HEAD has moved to Master's 1st commit (i.e. Commit number- 0aad0b9... in Figure-1.1).

.



```
Have fun!
$ git commit
$ git checkout e13
$ git checkout -b test
$ git commit
$ git merge master
```



```
Have fun!
$ git commit
$ git checkout e13
$ git checkout -b test
$ git commit
$ git merge master
$ git checkout HEAD^
```

Figure 1.2: Git graph before git checkout HEAD^ Git graph after git checkout HEAD^

Similarly in the 2nd case, on branch test we execute the command:- **git merge master**
Then we combined 2 branches together so our merge commit has 2 parents. So, on running the code:-
. **git checkout HEAD^**
We would find that HEAD has moved to test branch's 1st commit (i.e. Commit number- 719176d... in Figure-1.2).

Now we will see how git resolves this ambiguity

Caret ^ suggests an interesting segment of a tree or a fork in the road. The caret selector becomes useful with merge commits because each one is the child of two or more parents. In the other hand, Tilde ∼ is almost linear in appearance and wants to go backward in a straight line, favoring the first parent on merge commits.

Both ^ and ∼ on their own refer to the parent of the commit (^^ and ∼∼ both refer to the grandparent commit) But they have different meaning when they are used with numbers:^2 means the second parent where a commit has more than one parent because it's a merge.∼2 means up two levels in the hierarchy via the first parent if a commit has more than one parent.

Caret ^ is Branch selector, So **git checkout HEAD^2** selects the 2nd branch of a (merge) commit by moving onto the selected branch (one step backwards on the commit-tree). Whereas Tilde ∼ is Commit selector, So **git checkout HEAD∼2** moves 2 commits backwards on the selected branch.

Some Examples are shown below:-

We can specify one or more branches when merging. Then a commit has two or more parents and then ^ is useful to indicate parents. Suppose we are on branch B1 and we have two more branches B2 and B3 (Figure-1.3). On each branch the three last commits are:

B1: A, B, C
B2: P, Q, R
B3: X, Y, Z

If now on branch B1 we execute the command:- **git merge B2 B3**
Then we combined 3 branches together so our merge commit has three parents.

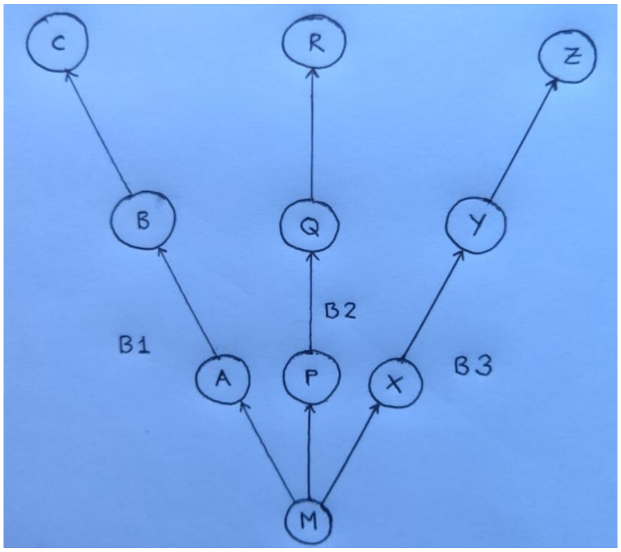


Figure 1.3: Git graph after git merge

We found-

^ indicates the n'th parent, while $\sim$ indicates the n'th ancestor in the first branch. Hence, the following result occured:-

HEAD^ indicates A
HEAD^2 indicates P
HEAD^3 indicates X
HEAD$\sim$ indicates A
HEAD$\sim$2 indicates B
HEAD$\sim$3 indicates C

1. HEAD$\sim$3 = HEAD$\sim\sim\sim$ = to HEAD^^^ (each indicating C)
2. HEAD$\sim$n = HEAD$\sim$ ... $\sim$ (n times $\sim$) = HEAD^...^ (n times ^).
3. HEAD^3 $\neq$ HEAD^^^ (HEAD^3 = X but HEAD^^^ = C)
4. HEAD^1 = HEAD^, but for $n > 1$ HEAD^n $\neq$ HEAD^...^ (n times ^) always.

One more Example is described below:-

Here is an illustration given of a sample git graph (Figure-1.4). Both commit nodes c2 and c3 are parents of commit node c1. Parent commits are ordered left-to-right.

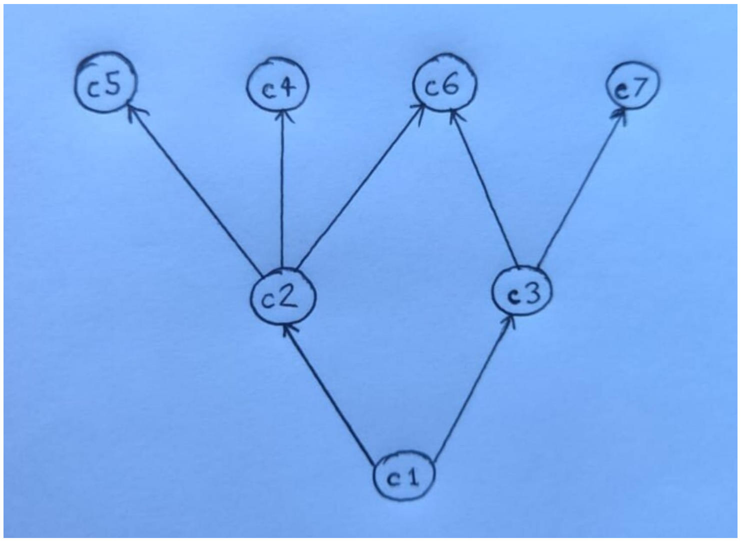


Figure 1.4: Git-graph Sample

We found-

$c1 = c1\hat{}0$
$c2 = c1\hat{} = c1\hat{}1 = c1 \sim 1$
$c3 = c1\hat{}2$
$c5 = c1\hat{}\hat{} = c1\hat{}1\hat{}1 = c1 \sim 2$
$c4 = c2\hat{}2 = c1\hat{}\hat{}2$
$c6 = c2\hat{}3 = c1\hat{}\hat{}3 = c1\hat{}2\hat{}$
$c7 = c3\hat{}2 = c1\hat{}2\hat{}2$

Thus in the two cases given in the question the HEAD always moves to the first parent i.e. last commit of the branch that **we merged into** (master in 1st case and test in the 2nd).

Reference Site:- https://stackoverflow.com/questions/2221658/whats-the-difference-between-head-and-head-in-git