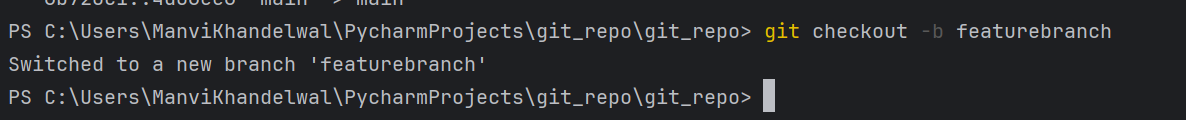
**Question 3**

Step 1: Create a feature branch.



Step 2: Switch to the new branch.

Step 3: open the file and make some changes to it.

Step 4: Add and commit the changes to the new branch.

A screen shot of a computer

Description automatically generated

Step 5: Push the changes to the new feature branch. A screen shot of a computer code

Description automatically generated

Step 6: Create a pull request.

A screenshot of a computer

Description automatically generated

Step 6: As another user in the master branch make some changes to the same file.

A screenshot of a computer

Description automatically generated

Step 7: Add and commit the changes to the master branch.

A screen shot of a computer

Description automatically generated

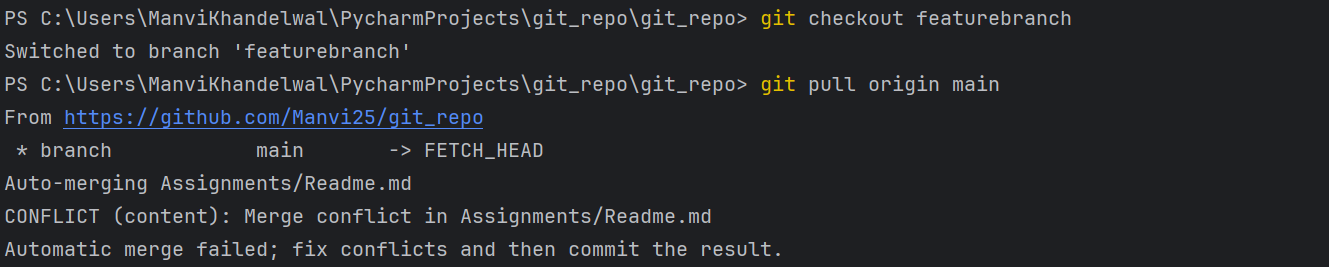
Step 8: Push the changes to the master branch. A computer screen shot of a program

Description automatically generated

Note: There will be a conflict in the pull request, how do we resolve it??

A screenshot of a computer

Description automatically generated



A screen shot of a computer code

Description automatically generated

A screenshot of a computer program

Description automatically generated

Hint: git rebase

**Question 4**

Step 1: Step 1: Create a feature branch.

Step 2: Switch to the new branch.

open the file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

Step 3: Identify the commit or commits that you want to "cherry-pick"(Note the hash of the commit or commits that you want to "cherry-pick")

Step 4: Use the "git checkout" command to switch to the branch where you want to apply the changes.

Step 5: Use the "git cherry-pick" command followed by the commit hash(es) that you want to apply.

**Question 5**

Step 1: Step 1: Create a feature branch.

Step 2: Switch to the new branch.

open the file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

open the same file and make some changes to it.

Add and commit the changes to the new branch.

Step 3: Use the "git log" command to view the commit history and identify the commit to which you want to reset.

Step 4: Use the "git reset" command followed by the desired reset type and the commit hash

Step 5: Verify that the reset was successful by using the "git log" command again.

Step 6: Use the "git log" command to view the commit history and identify the commit that you want to reverse.

Step 7: Use the "git revert" command followed by the commit hash or reference to which you want to revert. (Hint: git revert <commit hash>)

Step 8: Verify that the revert was successful by using the "git log" command again.

Note: Identify the difference between git log after git reset and git r evert.