**GIT Handson – 4**

* Explain how to resolve the conflict during merge.

**Solu –**

A merge conflict happens when Git can't automatically combine changes from two different branches into one. This usually occurs when the same line of code is edited in both branches. Git marks the conflicted file with special markers to show you where the problem is.

The conflict markers look like this:

<<<<<<< HEAD

This is the line from the current branch (HEAD).

=======

This is the line from the branch you're merging into.

>>>>>>> branch-name

* <<<<<<< HEAD: Marks the beginning of the changes from your current branch
* =======: Separates the changes from the two branches.
* >>>>>>> branch-name: Marks the end of the changes from the branch you're merging in

**Steps to Resolve a Conflict**

1. **Open the conflicted file:** Use your code editor to open the file that Git identified as having a conflict. You'll see the conflict markers mentioned above.
2. **Manually edit the file:** Go through the file and decide which changes you want to keep. You can:
   * Keep the changes from the current branch.
   * Keep the changes from the other branch.
   * Keep a combination of both.
   * Write a completely new version of the code.

After you've made your decision, delete the conflict markers (<<<<<<<, =======, >>>>>>>) and any code you don't want. The file should look like a normal, non-conflicted file when you're done.

1. **Add the resolved file:** Once you've manually fixed the file, you need to stage it to tell Git that you've resolved the conflict. Use the command git add <filename> for each file you've fixed. You can also use git add . to stage all resolved files.
2. **Commit the merge:** After all conflicts are resolved and the files are staged, you can commit the merge. Git will have already prepared a commit message for you, which you can edit or simply save. The command is git commit.