1.What is Git and why is it used?
Git is the most widely used version control system, enabling tracking of changes to files and easier collaboration among multiple users. Git can be accessed via a command line or through a desktop app with a graphical user interface

2.Explain the difference between Git pull and Git fetch.
The key difference between git fetch and pull is that git pull copies changes from a remote repository directly into your working directory, while git fetch does not. The git fetch command only copies changes into your local Git repo. The git pull command does both.

3.How do you revert a commit in Git?

1. Use the Git log or reflog command to find the ID of the commit you want to revert.
2. Enter the Git revert command (see below), including the commit ID you want to work on.
3. Provide an informative Git commit message to explain why you needed to perform the Git revert.

4.Describe the Git staging area.
The staging area is the middle ground between what you have done to your files (also known as the working directory) and what you had last committed (the HEAD commit). As the name implies, the staging area gives you space to prepare (stage) the changes that will be reflected on the next commit. This surely adds up some complexity to the process, but it also adds more flexibility to selectively prepare the commits as they can be modified several times in the staging area before committing.

5.What is a merge conflict, and how can it be resolved?
merge conflicts happen when people make different changes to the same line of the same file, or when one person edits a file and another person deletes the same file. You must resolve all merge conflicts before you can merge a pull request on GitHub.

Resolving a merge conflict on GitHub

- Under your repository name, click Pull requests.
- In the "Pull Requests" list, click the pull request with a merge conflict that you'd like to resolve.
- Near the bottom of your pull request, click Resolve conflicts.
- Decide if you want to keep only your branch's changes, keep only the other branch's changes, or make a brand new change, which may incorporate changes from both branches. Delete the conflict markers `<<<<<<<`, `=======`, `>>>>>>>` and make the changes you want in the final merge.
- If you have more than one merge conflict in your file, scroll down to the next set of conflict markers and repeat steps four and five to resolve your merge conflict.
- Once you've resolved all the conflicts in the file, click Mark as resolved.

- If you have more than one file with a conflict, select the next file you want to edit on the left side of the page under "conflicting files" and repeat steps four through seven until you've resolved all of your pull request's merge conflicts.
- Once you've resolved all your merge conflicts, click Commit merge. This merges the entire base branch into your head branch.
- If prompted, review the branch that you are committing to.
  If the head branch is the default branch of the repository, you can choose either to update this branch with the changes you made to resolve the conflict, or to create a new branch and use this as the head branch of the pull request.
  If you choose to create a new branch, enter a name for the branch.
  If the head branch of your pull request is protected you must create a new branch. You won't get the option to update the protected branch.
  Click Create branch and update my pull request or I understand, continue updating BRANCH. The button text corresponds to the action you are performing.
- To merge your pull request, click Merge pull request.

6.How does Git branching contribute to collaboration?

A common Git collaboration workflow is:

- Fetch and merge changes from the remote.
- Create a branch to work on a new project feature.
- Develop the feature on a branch and commit the work.
- Fetch and merge from the remote again (in case new commits were made)
- Push branch up to the remote for review.

7. What is the purpose of Git rebase?

The main aim of rebasing is to maintain a progressively straight and cleaner project history. Rebasing gives rise to a perfectly linear project history that can follow the end commit of the feature all the way to the beginning of the project without even forking. This makes it easier to navigate the project.

8. Explain the difference between Git clone and Git fork.

Forking creates your own copy of a repository in a remote location (for example, GitHub). Your own copy means that you will be able to contribute changes to your copy of the repository without affecting the original repository. Cloning makes a local copy of a repository, not your own copy.

9.How do you delete a branch in Git?

To issue the command to delete a local Git branch, follow these steps:

1. Open a Git BASH or a command prompt in the root of your Git repository.
2. If necessary, use the git switch or checkout command to move off the branch you wish to delete.
3. Issue the following command:
   git branch --delete <branchname>
4. Run the git branch -a command to verify the local Git branch is deleted.

10. What is a Git hook, and how can it be used?

Git hooks are scripts that run automatically every time a particular event occurs in a Git repository. They let you customize Git's internal behavior and trigger customizable actions at key points in the development life cycle.

Common use cases for Git hooks include encouraging a commit policy, altering the project environment depending on the state of the repository, and implementing continuous integration workflows. But, since scripts are infinitely customizable, you can use Git hooks to automate or optimize virtually any aspect of your development workflow.