**GIT Assignment 4**

**1Q.** How can new features be added to the main branch? What is a git conflict?

Ans.

Merge conflicts can be an intimidating experience. Luckily, Git offers powerful tools to help navigate and resolve conflicts. Git can handle most merges on its own with automatic merging features. A conflict arises when two separate branches have made edits to the same line in a file, or when a file has been deleted in one branch but edited in the other. Conflicts will most likely happen when working in a team environment.

There are many tools to help resolve merge conflicts. Git has plenty of command line tools we discussed here. For more detailed information on these tools visit stand-alone pages for git log, git reset, git status, git checkout, and git reset. In addition to the Git, many third-party tools offer streamlined merge conflict support features.

2Q. How do I remove a branch from GIT?

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

1. Open a [Git BASH window](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-customize-Git-Bash-Shell-prompt-settings) or Command Window 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

git branch --delete <branchname>

command to delete the local branch

1. Run the *git branch -a* command to verify the local Git branch is deleted

3Q. What is the purpose of the git checkout command?

Ans.

The git checkout command lets you navigate between the branches created by git branch. Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch. Think of it as a way to select which line of development you’re working on.

Having a dedicated branch for each new feature is a dramatic shift from a traditional SVN workflow. It makes it ridiculously easy to try new experiments without the fear of destroying existing functionality, and it makes it possible to work on many unrelated features at the same time. In addition, branches also facilitate several collaborative workflows.

The git checkout command may occasionally be confused with git clone. The difference between the two commands is that clone works to fetch code from a remote repository, alternatively checkout works to switch between versions of code already on the local system.

4Q. What is the purpose of the git commit command?

Ans.

While they share the same name, git commit is nothing like svn commit. This shared term can be a point of confusion for Git newcomers who have a svn background, and it is important to emphasize the difference. To compare git commit vs svn commit is to compare a centralized application model (svn) vs a distributed application model (Git). In SVN, a commit pushes changes from the local SVN client, to a remote centralized shared SVN repository. In Git, repositories are distributed, Snapshots are committed to the local repository, and this requires absolutely no interaction with other Git repositories. Git commits can later be pushed to arbitrary remote repositories.

5Q. What is the purpose of the command 'git rm'?

Ans. The git rm command can be used to remove individual files or a collection of files. The primary function of git rm is to remove tracked files from the Git index. Additionally, git rm can be used to remove files from both the staging index and the working directory. There is no option to remove a file from only the working directory. The files being operated on must be identical to the files in the current HEAD. If there is a discrepancy between the HEAD version of a file and the staging index or working tree version, Git will block the removal. This block is a safety mechanism to prevent removal of in-progress changes.

6Q. What is the purpose of the git log command?

# Ans. Git log

The advantage of a version control system is that it records changes. These records allow us to retrieve the data like commits, figuring out bugs, updates. But, all of this history will be useless if we cannot navigate it. At this point, we need the git log command.

Git log is a utility tool to review and read a history of everything that happens to a repository. Multiple options can be used with a git log to make history more specific.

Generally, the git log is a record of commits. A git log contains the following data:

* **A commit hash**, which is a 40 character checksum data generated by SHA (Secure Hash Algorithm) algorithm. It is a unique number.
* **Commit Author metadata**: The information of authors such as author name and email.
* **Commit Date metadata**: It's a date timestamp for the time of the commit.
* **Commit title/message**: It is the overview of the commit given in the commit message.

**7Q.** What is the purpose of 'git add'?

Ans. The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit. However, git add doesn't really affect the repository in any significant way—changes are not actually recorded until you run git commit.

In conjunction with these commands, you'll also need git status to view the state of the working directory and the staging area.

8Q. In GIT, what is a 'bare repository'?

Ans.

A bare git repository is intended to be used as a remote repository where code is shared between members of the team. The bare Git repo is not intended for local development.

After you issue a [git init bare command](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Quickly-create-a-git-bare-repo-with-init-or-clone), you won’t be able to write code in that repo, which also means you won’t have need to [clean up Git worktrees](https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/How-to-clean-up-Git-branches-and-commits) or workspaces.

9Q. What's the difference between git remote and git clone?

Ans. GIT REMOTE add just creates an entry in your git configuring  that specifies a name for a particular URL. You must have an existing git report  to use this.

GIT CLONE creates a new git repository by copying an existing one located at the URL you specify.

**The differences between git clone and git remote:**

**git clone:**

Will physically download the files into your computer. It will take space from your computer. If the repo is 200Mb, then it will download that all and place it in the directory you cloned.

**git remote add:**

Won't take space! It's more like a pointer! It doesn't increase your disk consumption. It just gets a snapshot of what branches are available and their git commit history I believe. It doesn't contain the actual file/folders of your project.

If you do:

git remote add TechLeadRepo git://github.com/user/test.git

then you haven't added anything to your computer. After you've added it in your remote branches then you're able to get a list of all branches on that remote by doing:

git fetch --all

upon fetching (or pulling), you download the files And then if you wanted to do get your colleague's feature22 branch into your local, you'd just do

git checkout -b myLocalFeature22 TechLeadRepo/feature22

Had you cloned his repo then you would have to go into that local repository's directory and simply just checkout to your desired branch