QUESTIONS-

1. Code for a project should be stored in\_\_\_\_\_\_\_\_\_\_\_\_.

A.Centralised System B.Distributed System

Explanation:In distributed systems data security and data consistency

Isn’t guaranteed.

1. What is the best way to store and share the changes made in a project?

A.Mail B.Dropbox/Google Drive

C.VCS D.Flash Drives

**About Version Control**

* Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later.
* If you are a graphic or web designer and want to keep every version of an image or layout
* TYPES-Local,distributed,centralised vcs

**GIT**

* Git is the industry-standard version control system for web developers
* Use Git commands to help keep track of changes made to a project.

Git has three main states that your files can reside in: modified, staged and committed:

* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit snapshot.
* Committed means that the data is safely stored in your local database.

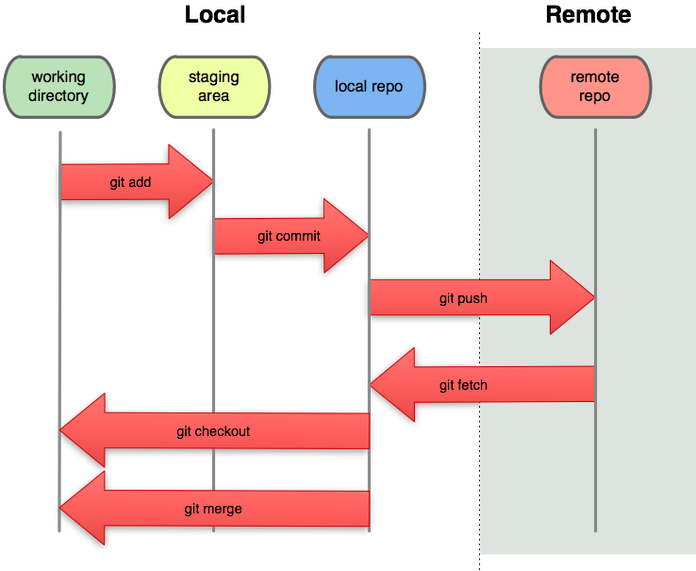
The basic Git workflow goes something like this:

* You modify files in your working tree.
* You selectively stage just those changes you want to be part of your next commit, which adds only those changes to the staging area.
* You do a commit, which takes the files as they are in the staging area and stores that snapshot permanently to your Git directory.



GIT COMMANDS-

* **git init-** creates a new Git repository
* **git status**- inspects the contents of the working directory and staging area
* **git add**- adds files from the working directory to the staging area
* **git diff** -shows the difference between the working directory and the staging area
* **git commit**- permanently stores file changes from the staging area in the repository



QUESTIONS-

1.GIT stores every change made in the system?T/F

2.Match the commands to their functions

* 1. A.init 1. permanently stores file changes
  2. B.add 2.Creates a new repository
  3. C.commit 3.adds files to staging area
  4. D.log 4.Diff btw file in staging area&working repository
  5. E.diff 5. Displays the log of all changes done

CODING-

1. Init a new git repository in a folder
2. Add file to staging area
3. Commit the changes
4. Add two files to the staging area using one command

* **MAKING CHANGES IN FILE**



QUESTIONS-

1. Is it compulsory to stage a document before commiting?
2. Can changes be made in working directory?
3. Can we add more than two files to the staging area at a time?

**BACKTRACKING-**

* One of the most useful features of any version control system is the ability to "undo" your mistakes. In Git, "undo" can mean many slightly different things.
* When you make a new commit, Git stores a snapshot of your repository at that specific moment in time; later, you can use Git to go back to an earlier version of your project.

**GIT RESET-**

This command is useful if you notice some small error in a recent commit (or set of commits) and want to redo that part without showing the undo in the history.

QUESTIONS-

1. Does GIT support backtracking?
2. Can GIT unstage all the file form staging area ?
3. Can we undo the changes in just one file without hampering other files that were commited together?

CODING-

Reset one file from staging area to working directory

**GIT PUSH-**

* Is used to push the data from local repository to remote repository
* Origin-the remote repository of git.
* Master-name of the current branch
* SYNTAX-git push origin master

QUESTION-

1. Can we push changes made in the same file from two or more different local repositories simultaneously?(try on your machine and then answer)
2. Is it compulsory to have an encryption key before uploading a document?

**GIT PULL -**

* It is used to pull the data from the remote repository.
* It only shows all the commited changes that are pushed on the remote repository done by others.
* git pull will append all the new commits from the remote repository on to your local repository one by one.
* pull **automatically merges the commits.**

**Resolving errors while pulling-**

In Git, there are two main ways to integrate changes from one branch into another: the merge and the rebase

1. **MERGE**

It performs a three-way merge between the two latest branch snapshots (C3 and C4) and the most recent common ancestor of the two (C2), creating a new snapshot (and commit).



Figure. Merging to integrate diverged work history

**2.REBASE-**

* There is another way: you can take the patch of the change that was introduced in C4 and reapply it on top of C3 this is called rebasing.



Figure. Rebasing the change introduced in C4 onto C3

QUESTIONS-

1. What commands can be used to integrate changes made in two different branches?( In Git, there are two main ways to integrate changes from one branch into another: the merge and the rebase)
2. What amongst merge and rebase is better according to you?

GIT REBASE-

* With the rebase command, you can take all the changes that were committed on one branch and then apply the changes of another branch on top of it.

QUESTION-

* 1. What command would you use to pull changes from the remote server using rebase strategy?

CODING-

1. Try pushing changes made in the same file from two or more different local repositories simultaneously.

(solution:use rebase)

Ways to resolve rebase conflicts-

1.GIT rebase --continue

* we can avoid conficts by resolving all the issues manually and then adding the files and then indicate that all issues are resolved by using this command.
* The programmer would be responsible for any unresolved conflict

2.GIT rebase --skip

* To skip the commit and get back to the state before rebase.it shouldn’t be ideally used as we can lose all the changes done in previous commit.

3.GIT rebase --abort

* To get back to the original branch.