A. Answer following questions briefly:

1. List out the key difference between a centralized version control system and distributed version control system
2. List down any two centralized version control system and 2 distributed version control system
3. What are the advantages of git VCS over other VCS
4. What are the different states of a file in the Git VCS

**B. GIT REMOTE REPOSITORIES:** Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.

1. Clone an existing repository on Github created during course and configure your local repo to point to the remote repository
2. Perform some operation like add, remove, modify and finally push your changes to the remote repository
3. Pull the latest changes from the repository to get the updates from others in to your local repo and merge the changes
4. Try fetching the changes and perform the merge to get the difference between the pull and the merge command
5. Perform some changes and before committing the changes, stash your changes and then pull the changes and finally apply the changes to understand how stashing works
6. Create a feature branch and do some file operations in the branch and commit the changes to the branch
7. Merge the changes using the rebase command and finally perform a safe deletion of the feature branch
8. Create another feature branch and this time after committing the changes to the feature branch, merge the changes using fast forward merge and then delete the feature branch

**C. GIT BRANCHING AND MERGING:** Perform following tasks and explain how you performed each operation. Draw a flow diagram as you progress through the steps. Add all git commands you used and push the repository in your github. Add your github public repository (for these following tasks) link with the homework.

1. Create a local branch using git checkout -b branchname command
2. Observe the difference by doing some file operations and switch back to the master branch and see if you can see the changes done on the branch
3. Now switch back to the branch and commit the changes and switch to master

branch. Now see if you can still see the changes in the master branch

1. Now switch back to the branch name and stash the changes and apply the changes to the master branch by switching to the master branch
2. Try merging the changes from the branch to the master branch using all the three merge strategies and then view the git log
3. Push the local branch to the remote repository and see if the branch is present on the remote repository – Github

**D: Review following “MERGE with CONFLICT” scenario. Complete the exercise as requested below:**

**Purpose:**

Learn how to merge when there are code conflicts

Learn how to interact with the remote repository

**Preparation**

**Watch following video:** [**https://vimeo.com/138418055**](https://vimeo.com/138418055)

**You** may want to download the script used for video**. (copied at the end of this assignment)**

**Exercise**

**For t**his exercise, you should experiment with the merging files that have conflicts. You may use the code below or an*y* code of your choice.

public class TheMotivator {

public void feedback(int score) {

if (score == 100)

System.out.println("You're awesome");

else if (score > 90)

System.out.println("That's great");

else if (score > 60)

System.out.println("That's good ");

else

System.out.println("Well, what can I say?");

}

public static void main(String[] args) {

TheMotivator tm = new TheMotivator();

tm.feedback(60);

}

}

**Specific Requirements**

**Th**is exercise is worth 5 points. Turn in a git log that contains at least two different branches that have been merged. NOTE: The point of this is to get comfortable with git, so you do not need to follow the demo exactly. The log should show some reasonable amount of branching and merging. You should do some merges with and without conflicts, although we won't be able to tell this from the log. Show with diagram also.

Remember that to create log file you do:

git log --pretty=format:"%s" --graph > mylog.txt

I would suggest you spend at least half an hour on this task (more if this is all new to you).

**Submit**

Explain steps followed, draw the git flow diagram and

Submit your .txt log file “mylog.txt” with you assignment.

**Hint: Script used in the video**: [**https://vimeo.com/138418055**](https://vimeo.com/138418055)

Git Merging Demo Steps

Create SomeClass

Add method fnOne [syso: I couldn't repair your brakes, so I made your horn louder]

Run

git bash

cd to directory

git init

git status

git add \*.java

git status

git commit -m "Initial"

chg fnOne to thoughtForTheDay

git status

git commit -a -m "Refactored fnOne"

git checkout -b addEvents

add upcomingEvents [syso: Party at Jane's house tomorrow]

git commit -a -m "Added events"

add header to upcomingEvents

try: git checkout master, see error

git commit -a -m "Refined events"

now: git checkout master

revise thought for the day: On the other hand, you have different fingers

git commit -a -m "New thought for the day"

git branch --no-merged (see addEvent)

git merge addEvents [ success! different parts of the file]

git branch --no-merged

git branch --merged

git checkout -b moreEvents

Modify program:

variable:

private ArrayList<String> events = new ArrayList<String>();

new method:

public void createEvents() {

events.add("We're going to a movie on Saturday");

events.add("Study session on Sunday - Jim's house");

}

change method:

public void upcomingEvents() {

System.out.println("Upcoming Events");

for (String event : events)

System.out.println(event);

}

call in main:

sc.createEvents();

git commit -a -m "Add multiple events"

git checkout master

modify upcomingEvents ["Dinner at Katie's on Friday"]

add SomeClass sc = new SomeClass() to main

git commit -a -m "Different event"

git merge moreEvents [ conflicts! need to resolve]

in Editor, notice the lines with issues, fix!

>> how? remove lines from head, remove lines with === and <<<

>> in general? first decide which to keep, make these kinds of

>> changes.

git branch --no-merged

git merge [ won't let you yet!]

git commit -a -m "Merged event handling"

git branch --no-merged

git branch --merged

git checkout moreEvents

modify: Upcoming Events - Please join us!

git commit -a -m "More friendly events"

git checkout master

press up-arrow, git branch --no-merged

git merge moreEvents [success! it's only a conflict if 2 changes]

git log

git log -p -2

git log --pretty=oneline

git log --pretty=format:"%s" --graph

git log --pretty=format:"%s" --graph > mylog.txt

git config --global alias.gr 'log --pretty=format:"%s" --graph'