**Version Control**

Item 11 has all the links for GitHub

git help -a *and* git help -g *list available subcommands*

touch  *creates a new file*

mkdir  *creates a new folder*

mv  *changes a file or folder to something else i.e. mv test.txt test123.txt (the first file is changed*

*to the second)*

rm  *removes files*

rm -r  *removes directories (the -r is called a switch command)*

ls -a  *lists everything including hidden files*

ls -la  *lists everything including details and permissions*

(program to run) --overwrite --dry-run (directory to be overwritten) *lists all files that would be*

*deleted.*

cat *when followed by a file name will print the text contained in that file to the console*

grep  *search a file for an existence of a specific word or phrase*

git checkout -b (name of branch) *allows the use of a branch that hasn’t been created yet*

git add -A *adds all changes made to all files in a directory*

git shortlog *sorts output by author* -n *declares how many commits per author*

**Tutorial Commands:**

Git ls *Shows directories and files within current directory*

mkdir *Creates a folder*

touch *Creates a file*

git mv originalName.file newName.file *Changes name of file*

git rm name.file *Removes a file*

git rm -r *Removes a directory*

git init *Activates a repository to watch for changes to files in the directory*

git add name.file *Adds files to repository*

git commit -m "Put description of change here for future reference"

git commit -a -m "Description of change for multiple files"

git status *displays current status of changes and commits*

git log *Shows a history of commits*

git branch *Shows all available branches*

git branch branch2 *Creates a branch named branch2*

git checkout branch2 *Causes changes to be made to branch2, but other branches remain unaffected*

git branch -D branchName *Deletes branchName branch*

git merge branch2 *Merges branch2 to branch that is currently active*

git

git

git

**FILE LOCATIONS:**

Bash completion has been installed to:

 /usr/local/etc/bash\_completion.d

Emacs Lisp files have been installed to:

 /usr/local/share/emacs/site-lisp/git

Version control system VCS - Provides two capabilities 1) lock files so they can only be edited by one person at a time by allowing users to “check out” files, and 2) track changes to files.

Local - (Saves files to a database on one computer)

Centralized - One central repository for all users, con one server if it goes down

(Saves changes to a shared server)

Distributed - Uses one local hard drive on the developer system.

(Allows for more access than a local version control system, and isn’t as susceptible as a centralized version control system to loss of data if the server goes down.)

VCS saves only the changes not complete copies of each new version

Github - uses copy-modify-merge method making it possible for hundreds of developers to be working at the same time.

Without version control programs are written by coding cowboys

Working set - files that you want to add to the master repository this is done by doing a commit.

Pull requests initiate discussion about commits. Fork & Pull Model you notify the project maintainers of changes you would like them to consider. With Shared Repository Model a pull request starts review process and conversation about proposed changes before a commit takes place.

-By using @mention you can ask for feedback from specific people or teams

Git - refers to the implementation of vcs

**Essay #1**

Version control - Allows developers to use past versions of a program if something isn’t working in the current version and identify what changes caused it to stop working.  
VCS - A version control system offers the ability for multiple developers to work on the same project by checking out a file and editing it then checking it back in. This can be done in a centralized (utilizing a central server) or distributed (all files on one hard drive) method.  
TFS - A team foundation server is a centralized type of VCS, and is used to let large groups of people contribute to a project in a fluid ongoing way.  
Branch - The program starts with a master branch and then when a change is made to a file it creates a new branch off of the master branch.  
Merge - A merge happens when side branches are integrated back into the master branch.

**Essay #2**

What is Git?

Git is a version control system program that offers the ability to easily change and track changes made to a file. Git offer a level of control at the master branch, so when a developer pushes a change it must be accepted, this encourages a more efficient change flow. It works quickly and helps to take the focus off of the VCS and put the focus on the people, contributions, and conversations.

What is GitHub?

GitHub is the user interface that utilizes Git. Git is the command line and GitHub is the GUI. Github is also the place where multiple developers using Git connect and collaborate.   
How can you use these tools as a software developer?

These tools are essential to software development, because they can accept input from multiple people working on the same project and allow developers to track changes and revert back to a previous version if the current one isn’t working. These tools also incorporate both centralized and distributed methods of accepting input for a more user friendly experience.

What is version control?   
How does Distributed Version Control differ from Centralized Version Control?   
What benefits are there in understanding Git, GitHub and version control in general?   
How can you use these tools as a software developer?

Version control is a system that is designed to keep track of all of the changes made to the files and folders (directories) that are part of a project. It allows users the ability to revert back to an earlier version and identify the problem if something goes wrong with the current version. Version control systems are also set up to take contributions from multiple users that can be managed by a project leader.  
The two main categories of version control are distributed and centralized and the difference between them is that with a centralized system there is one master version of the project and users contribute to specific pieces of the project and then when the changes are verified they are added to the master, with a distributed system the entire project is copied and then distributed to a user so that they can make changes and when they want to add it back to the original copy it is verified and then merged.

There are several benefits to understanding Git and GitHub. Git allows a user to make changes on their own computer and GitHub is a very functional site for compiling, verifying, and viewing the changes that were made by various users.

These tools are an integral part of a software developer’s tool box. They can be used to keep every project organized and tracked. They also help to declutter the users computer by not having to keep multiple copies of files at different stages on their own hard drive.