CSPB 3308 Summer 2024 - Nath - Software Development Methods and Tools

Started on	Thursday, 13 June 2024, 7:45 PM
State	Finished
Completed on	Thursday, 13 June 2024, 7:50 PM
Time taken	5 mins 31 secs
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 2.00 out of 2.00

Please match the following git commands with the best description of the function/purpose of that command.

git checkout	moves the HEAD pointer to a different branch making that branch active ✔
git diff	compares two versions of a git-managed file ✔
git log	shows a record of recent commits ✔
git init	creates an empty local repository ✔
git add	stages a file under git tracking making it ready for a commit ✔
git commit	copies a staged file into the local git repository ✔

Your answer is correct.

The correct answer is: git checkout → moves the HEAD pointer to a different branch making that branch active, git diff → compares two versions of a git-managed file, git log → shows a record of recent commits, git init → creates an empty local repository, git add → stages a file under git tracking making it ready for a commit, git commit → copies a staged file into the local git repository

Question **2**

Correct

Mark 1.00 out of 1.00

The Git **clone** command does which of the following?

- a. Creates a working directory
- b. Makes a local copy of the repository
- c. Commits a new branch
- d. a and b ✔
- e. a, b, and c

Your answer is correct.

The correct answer is: a and b

Question **3**

Correct

Mark 1.00 out of 1.00

Git needs to connect to a server to commit changes.

Select one:

- True
- False ✔

Correct. The Git will **commit** to your local repository and you would use **push**to copy your local commits to a remote server

Git is distributed, there is no central repository....

The correct answer is 'False'.

Question **4**

Correct

Mark 1.00 out of 1.00

git uses a hash to uniquely identify commits.

Select one:

- True ✔
- False

You can see the commit comments and the unique identifier for each commit using the git log command.

The correct answer is 'True'.

Question **5**

Correct

Mark 1.00 out of 1.00

After you initialize your repository with `git init`, git automatically tracks every file in the repository.

Select one:

- True
- False ✔

You have to tell git which files to track.

The correct answer is 'False'.

Question **6**

Correct

Mark 1.00 out of 1.00

The command `patch` patches bugs in your files.

Select one:

- True
- False ✔

Watch the first video about version control.

The correct answer is 'False'.

Question **7**

Correct

Mark 1.00 out of 1.00

main is a special branch marking where you are right now.

Select one:

- True
- False ✔

When working with Git, only one branch can be checked out at a time. **HEAD** is the special name that points to the "active" or "current" branch. The **main** branch is usually used is the "main" branch or trunk of the tree. All other branches will eventually flow back into the **main** branch.

The correct answer is 'False'.

Question **8**

Correct

Mark 1.00 out of 1.00

Now, imagine that you have a local repository, but other team members have pushed changes into the remote repository. What Git operation would you use to download those changes into your working copy?

- a. checkout
- b. pull ✔
- c. export
- d. import
- e. update
- f. commit

Your answer is correct.

The correct answer is: pull

Question **9**

Correct

Mark 1.00 out of 1.00

Why would you use a git **branch**?
Select all the appropriate reasons below.

Select one or more:

- [x] a. Isolation of work: By working on a separate branch, you can make changes to your code without affecting the main codebase, allowing you to experiment and try out new ideas without risking breaking anything. ✔
- [ ] b. Different Projects: to keep the project code separated from code for other projects, each branch would support a different project.
- [x] c. Bug fixes: You can create a new branch to fix a bug and then merge that branch back into the main branch, without affecting the other development that is going on. ✔
- [ ] d. Creating Backups: create branches to keep each version of the code separated from development by other project members.
- [x] e. Feature development: You can create a new branch to develop a new feature, test it and merge it back to the main branch after it is completed. ✔
- [x] f. Release management: Branches can be used to manage different versions of a codebase. For example, you might have a "development" branch where new features are added, and a "release" branch where only stable versions of the code are kept. ✔
- [x] g. Collaboration: Branches can be shared with other developers, who can then make their own changes and submit them back to the main branch via a pull request. This makes it easy to collaborate on a project without stepping on each other's toes. ✔
- [ ] h. Personal Version: Keeping a local copy of the repository for development outside of the project that others will not change.

Your answer is correct.

Isolation of work: By working on a separate branch, you can make changes to your code without affecting the main codebase, allowing you to experiment and try out new ideas without risking breaking anything.

Collaboration: Branches can be shared with other developers, who can then make their own changes and submit them back to the main branch via a pull request. This makes it easy to collaborate on a project without stepping on each other's toes.

Release management: Branches can be used to manage different versions of a codebase. For example, you might have a "development" branch where new features are added, and a "release" branch where only stable versions of the code are kept.

Bug fixes: You can create a new branch to fix a bug and then merge that branch back into the main branch, without affecting the other development that is going on.

Feature development: You can create a new branch to develop a new feature, test it and merge it back to the main branch after it is completed.

The correct answers are: Isolation of work: By working on a separate branch, you can make changes to your code without affecting the main codebase, allowing you to experiment and try out new ideas without risking breaking anything., Collaboration: Branches can be shared with other developers, who can then make their own changes and submit them back to the main branch via a pull request. This makes it easy to collaborate on a project without stepping on each other's toes., Release management: Branches can be used to manage different versions of a codebase. For example, you might have a "development" branch where new features are added, and a "release" branch where only stable versions of the code are kept., Bug fixes: You can create a new branch to fix a bug and then merge that branch back into the main branch, without affecting the other development that is going on., Feature development: You can create a new branch to develop a new feature, test it and merge it back to the main branch after it is completed.