## Question - 1
**Git: Feature Removal from a Development Branch**

SCORE: **50 points**

DevOps | Git | Easy | Git Basic Commands | Git Log | Git Branching

In software development, managing features through version control is critical to project management and workflow efficiency. Sometimes, a feature must be removed, either because it is no longer needed, deprecated, or being reworked. This task involves working with branches in Git to ensure that changes are made in the appropriate context without affecting the stable version of the project.

For this task, within an existing project repository, the objective is to remove a specific feature (represented by the file "feature.module") in a development branch that will be created from the current master branch. This approach allows for isolated development and testing without impacting the master branch. After the feature removal, the changes should be committed to the development branch with a specific commit message and then pushed to the remote repository.

Using the existing project repository located at `/home/ubuntu/1792332-git-feature-removal-from-a-development-branch`:

- Check out the current master branch to ensure it is up to date.
- Create a new branch named "development" from the master branch.
- In the "development" branch, remove the file "feature.module".
- Commit the removal of "feature.module" with the commit message "REQUEST-15: Remove feature".
- Push the "development" branch to the origin remote repository as "development".

An example of the desired output of `git status`:

```
On branch development
Your branch is up to date with 'origin/development'.

nothing to commit, working tree clean
```

An example of the desired output of `git log --name-status`:

```
commit 5a89545f744f25a72328d31cf494b4048dd701cd (HEAD -> development, origin/development)
Author: Hacker Developer <hacker.developer@hackercompany.com>
Date:   Sun Jun 9 16:23:10 2024 +0000

    REQUEST-15: Remove feature

D   feature.module

commit 0f6f2c1905ac801c92a4c1efe7ce1e4a2dc859e8 (origin/master, master)
Author: Hacker Developer <hacker.developer@hackercompany.com>
Date:   Sun Jun 9 16:18:17 2024 +0000

    FEATURE-1: Feature implementation

A   feature.module

commit df5e78cf6d4d934e81bf02c1c5f1c4e6100f48a2
Author: Hacker Developer <hacker.developer@hackercompany.com>
Date:   Sun Jun 9 16:18:17 2024 +0000

    TASK-1: Project initialization

A   core.module
```

## Question - 2

SCORE: **50 points**

**Git: Amending the Last Commit Message**

DevOps | Git | Easy | Git Basic Commands | Git Log | Git Amend

In the realm of software development, especially within projects utilizing continuous integration and continuous delivery (CI/CD) systems, the structure and content of commit messages can be critical. Certain CI/CD pipelines are configured to trigger or skip steps based on specific keywords or identifiers within commit messages. Occasionally, a commit may be pushed without the necessary identifiers, necessitating a quick correction to ensure the integrity of the automated workflows.

For this task, within an existing project repository, the objective is to amend the last commit message by appending a specific task ID that was inadvertently omitted. This correction is vital for maintaining the seamless operation of the CI/CD pipeline.

Using the existing project repository located at `/home/ubuntu/1792055-git-amending-the-last-commit-message`:

- Identify the last commit message.
- Amend the last commit message to append the task ID, transforming the format from "Module A implementation" to "TASK-2: Module A implementation".
- Push the amended commit to the remote repository, taking care to preserve the continuity of the project history and the CI/CD pipeline.

An example of the desired output of `git status`:

```
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

An example of the desired output of `git log --name-status`:

```
commit 92c3efc596fa3f8ec7a8c142158a18d2b7718184 (HEAD -> master, origin/master)
Author: Hacker Developer <hacker.developer@hackercompany.com>
Date:   Sun Jun 9 12:01:54 2024 +0000

    TASK-2: Module A implementation

A   a.module

commit db5ef16f152eb1000b33e36701748c5265148678
Author: Hacker Developer <hacker.developer@hackercompany.com>
Date:   Sun Jun 9 12:01:54 2024 +0000

    TASK-1: Project initialization

A   core.module
```

## Question - 3

SCORE: **50 points**

**Git: Repository Initialization and First Commit**

DevOps | Git | Easy | Git Basic Commands | Git Config | Git Log

In the early stages of a software development project, setting up a version control system is crucial for tracking changes, collaborating with team members, and maintaining the project's overall integrity. Git, a distributed version control system, offers a robust framework for managing a project's evolving codebase.

For this task, the goal is to initialize a new Git repository, configure user identity, add all existing files in the project to the repository, and make the first commit.

Using the local project directory named `/home/ubuntu/1792002-git-repository-initialization-and-first-commit`:

- Initialize the directory as a Git repository.
- Configure the Git user name and email.
- Add all files and folders in the directory to the staging area, preparing them for the first commit.
- Commit the added files with the commit message "Initial project setup".

An example of the desired output of `git status`:

```
On branch master
nothing to commit, working tree clean
```

An example of the desired output of `git log --name-status`:

```
commit e978b16248473a2a7c05c2f32306d626baa1c099 (HEAD -> master)
Author: User Name <user.email@domain.example>
Date:   Sun Jun 9 06:28:14 2024 +0000

    Initial project setup

A    README.md
A    core.module
A    modules/a.module
A    modules/b.module
```

## Question - 4

SCORE: **50 points**

**Git: Initial Commit**

Git | DevOps | Git Config | Git Add | Git Commit | Git Push | Easy

You are working on the application development process and need to commit your changes to Git.

Using the existing Git repository "/home/ubuntu/1326005-git-initial-commit":
- Set up a Git username at the repository level (not globally!), set it to "Hacker Developer".
- Set up a Git email address at the repository level (not globally!), set it to "hacker.developer@hackercompany.com".
- Commit all available files in "/home/ubuntu/1326005-git-initial-commit" directory to the Git repository with the message "Initial implementation"
- Push all to the remote origin

**Note:**
- The completed solution will be evaluated in a new, clean environment. ONLY CHANGES IN "/home/ubuntu/1326005-git-initial-commit" WILL BE CARRIED TO THE NEW ENVIRONMENT. ALL CHANGES OUTSIDE THIS DIRECTORY WILL BE LOST.

## Question - 5

SCORE: **50 points**

**Git: Setup User Identity**

Git | DevOps | Easy | Git Config

You are working on the application development process and need to set up your Git identity.

Using the existing Git repository "/home/ubuntu/1325993-git-setup-user-identity":
- Set up a Git username at the repository level (not globally!), set it to "Hacker Developer".
- Set up a Git email address at the repository level (not globally!), set it to "hacker.developer@hackercompany.com".

**Note:**
- The completed solution will be evaluated in a new, clean environment. ONLY CHANGES IN "/home/ubuntu/1325993-git-setup-user-identity" WILL BE CARRIED TO THE NEW ENVIRONMENT. ALL CHANGES OUTSIDE THIS DIRECTORY WILL BE LOST.

## Question - 6

SCORE: **5 points**

**git status**

Git Easy

Based on the output of the 'git status' command that is shown, which of the following statements is true?

```
git status
On branch newbranch
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:  README.md

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:  README.md
```

- ○ There are 2 files with the name README.md.
- ○ `git reset --soft HEAD~` was executed and the last commit added the REAMDE.md file.
- ○ `git reset --mixed HEAD~` was executed and the last commit added the REAMDE.md file.
- ○ `git reset --hard HEAD~` was executed and the last commit added the REAMDE.md file.
- [✓ `git add README.md` was executed and then another change was made to the README.md file.]

## Question - 7

SCORE: **5 points**

**git describe**

Git Easy

`git describe <commit-id>` describes the commit against

- ○ the HEAD.
- ○ the nearest branch.
- [✓ the nearest tag.]
- ○ the latest commit of the main branch.

## Question - 8

SCORE: **5 points**

**git squash**

Git Easy

Which command(s) will squash the last 3 commits into one commit? Pick all that apply.

- [✓ git rebase -i HEAD~3 ; In the screen that opens up, change “pick” to “squash” for the 2 commits before the latest commit.]
- [✓ git rebase -i HEAD~3 ; In the screen that opens up, change “pick” to “fixup” for the 2 commits before the latest commit.]
- [✓ git reset --soft HEAD~3 && git commit]
- git squash 3

## Question - 9
**Clone a Git Repo**

**SCORE:** 5 points

Git Easy

Which command(s) will clone a Git repository with submodules? Select all that apply.

- [✓ git clone --recurse-submodules https://github.com/cameronmcnz/surface.git]
- git clone --recursive https://github.com/cameronmcnz/surface.git
- [✓ git clone https://github.com/cameronmcnz/surface.git ; git submodule init; git submodule update --recursive]
- git clone https://github.com/cameronmcnz/surface.git; git fetch --all --recursive.
- [✓ git clone https://github.com/cameronmcnz/surface.git ;git submodule update --init --recursive]

## Question - 10
**How Many Branches?**

**SCORE:** 5 points

Git Easy

You have just cloned a remote repository to your local disk. The remote repository has 5 branches. How many branches does your local repository have?

- 5
- [✓ 6]
- 10
- 15

## Question - 11
**Git Languages**

**SCORE:** 5 points

Git Easy

Which language(s) are used to build git?

- Java
- [✓ C]
- [✓ Perl]

[✓ Python]

Swift

## Question - 12
**Git Data Structure**

SCORE: **5 points**

Git Hard Hard

Which data structure is git modelled after? In other words, when you look under the hood, which data structure does git resemble?

- a queue
- a linked list, every commit links to its parent
- a stack
- [✓ a hash or a key-value store]
- an array

## Question - 13
**Remove Sensitive Info**

SCORE: **5 points**

Git Hard

You have accidentally pushed a passwords.txt to your git repository. How do you remove this password from the git repo safely, so it can never be seen by anybody else. Select all correct answers.

- Execute `git reset --hard HEAD~` and do a force push, which removes the commit from the repo.
- Revert the commit using `git revert` and git push.
- [✓ Use the tool BFG repo cleaner.]
- Use the git branch-filter command to remove unwanted data.
- [✓ Use the git filter-repo tool]

## Question - 14
**HEAD^2**

SCORE: **5 points**

Git Hard Hard

Consider the output of the git log command.

```
*   dcef518 (HEAD -> master, newbranch) Merge branch 'newbranch'

|\

| * 0824c9e adding index.html

* | 6c50d08 adding about.html
```

```
|/

* 80be5ff new commit to newbranch

* 98bf46c Ading more content

* 00b3edf Initial Commit
```

On this repo, which commit is checked out when the following command is executed?

```
git checkout HEAD^2
```

- ○ 80be5ff
- ○ 98bf46c
- ○ 00b3edf
- [✓ 0824c9e]
- ○ dcef518

## Question - 15

**Delete a Remote Branch**

SCORE: **5 points**

Hard Git

Which of the following commands can you use to delete a remote/upstream branch *hotfix1* in git? Select all that apply.

- ○ git branch -D hotfix1
- [✓ git push origin --delete hotfix1]
- [✓ git push origin :hotfix1]
- [✓ git push origin -d hotfix1]
- ○ git branch --delete hotfix1

## Question - 16

**HEAD~5**

SCORE: **5 points**

Git Hard Hard

Consider the output of the git log command.

```
*   dcef518 (HEAD -> master, newbranch) Merge branch 'newbranch'

|\

| * 0824c9e adding index.html

* | 6c50d08 adding about.html
```

```
|/

* 80be5ff new commit to newbranch

* 98bf46c Adding more content

* 00b3edf Initial Commit
```

On this repo, which commit is checked out when the following command is executed?

```
git checkout HEAD~5
```

- ○ 80be5ff
- ○ 98bf46c
- ○ 00b3edf
- ○ 0824c9e
- [✓ dcef518]

## Question - 17
**Branch Names**

SCORE: **5 points**

Git Easy

Which of the following are valid branch names?

- ○ new..branch
- ○ new branch
- [✓ new_branch]
- [✓ new/branch]
- ○ .newbranch
- ○ /new_branch

## Question - 18
**Git Blame**

SCORE: **5 points**

Git Hard

Which statements are true about git blame?

- [✓ Identifies a particular commit.]
- [✓ Tells us who modified each line in a file and which commit was responsible for the changes.]
- ○ Shows the lines that were deleted or replaced.

○ All of these.

## Question - 19

SCORE: **5 points**

**File Commits**

Git  Hard

You have recently committed the file *LearningGit.c.* This is your 7th commit. Find the difference between your 2nd and 4th commit. Choose all that apply.

○ git diff head^^ head^^^^ LearningGit.c

○ git diff head~2 head~4 LearningGit.c

[✓ git diff head~5 head~3 LearningGit.c]

[✓ git diff head^^^^^ head^^^ LearningGit.c]