Question 01: Briefly explain the difference between git and GitHub? What is the relationship between them? (/3 marks)

Git is a version control system. It is used to keep a history of the user's changes on local repository. GitHub is a cloud based hosting service which is used to keep track or manage git repositories. Programmers can construct continuous coding projects without worrying about breaking anything thanks to the helpful version-control features provided by both Git and GitHub. By providing more features and tools, as well as a location online to store and collaborate on projects, GitHub simply goes a little bit farther than Git.

Question 02: What are three advantages of version control systems (VCS)? (/2 marks)

- The document can progress more effectively thanks to version control. Teams strive to make difficult procedures simpler so that there is more room for automation and consistency. So it improves efficiency
- The user can trace back to changes he/she made if there's something wrong with the current version and can check where it went wrong
- It provides a history of changes, who made the changes, date etc.
- It allows multiple users to work on the project concurrently. So it's easier for collaboration
- Creating branches, merge allows the user to test with other features or add additional feature without impacting the original version

Question 03: Complete the following table outlining a series of common git Commands. You should briefly describe the purpose of each command in the area provided. (/5 marks)

Clone: Git clone is usually used to create a copy or clone of an existing repository in a new directory at a different location.
Add: A modification in the working directory is added to the staging area using the git add command. It instructs Git that you wish to add updates to a certain file to the upcoming commit. Changes aren't truly logged until you execute git commit, thus git add doesn't really have any impact on the repository.
Commit: The project's staged modifications are captured in a snapshot via the git commit command. It's a "safe" version of a project because Git won't modify them until the user push the changes into the git repository.
Push: To upload content from a local repository to a remote repository, use the git push command. Commits are sent from the user's local repository to a remote repository via pushing.
Pull: Using the git pull command, you may quickly update your local repository with the content you just downloaded from a remote repository. So if the user has an old version of the repository, it will update it to the new one from the cloud repository.

Question 04: What is the purpose of branching in git? (/ 3marks)

Essentially, it's a way to make changes in the code without affecting the original version. It is used for adding new features or if someone wants to play around to make some changes and see how it will affect the project. If the feature is successfully added or the changes made were satisfactory, the user will have the option to merge the changes into the original branch.

Question 05: Assuming you have created a new feature in branch, how do you merge that branch back into your master? (/3 marks)

The branch can be merged back into the main code branch after the feature is finished. To switch the active branch back to the master branch, we first execute git checkout master. The new feature is then merged into the master branch using the git merge new-branch-name command. The main branch will now reflect the new commits. However, Git might be unable to finish the merge because of conflicting change in the source branch. That’s called merge conflict.

Question 06: Please explain what is meant by the term “merge conflict” in git. Try and provide an example where such an incident might occur. (/3 marks)

When the source and target branches of a merge request contain different changes, a merge conflict occurs, and you must choose which change to accept. Git analyses the two versions of the files line by line when a merge request is made. GitHub can typically merge changes together. GitHub prevents a merging when two branches both modify the same line, thus you must decide which modification to maintain.

An example can be when one person has edited a file and another person deleted the file or made changes on the same line, merge conflict can occur.

Question 07: Relating to question 06, what is the common (or manual) way of resolving a “merge conflict”? You can explain the main idea or use git commands. (/3 marks)

The user can go line by line for checking the changes and decide which change he/she wants to keep and resolve the conflicting edits. If the file is deleted by the other user, the current user can decide if he wants to keep the file or want to delete and then push his branch.

Question 08: An alternative to “merge” is “rebase”. Briefly research and explain the differences between “merge” and “rebase” operations in git. (/4 marks)

Merge	Rebase
You can join Git branches with the command git merge.	Developers can incorporate modifications from one branch to another using the command "git rebase."
The entire history of the merging of commits will be displayed in the Git Merge logs.	As commits are rebased with Git rebase, logs are linear.
The master branch will consolidate all of the feature branch's commits into a single commit.	The master branch will receive the same amount of commits once each commit is rebased.
When the destination branch is a shared branch, Git Merge is used.	When the destination branch is the private branch, Git Rebase should be used.

Question 09: What is a “pull request”? Explain the process of using a “pull request” to submit changes from a feature “branch”. (/3 marks)

When a contributor or developer is prepared to start the process of integrating new code changes with the main project repository, they submit a pull request, also known as a merge request.
When a developer submits a pull request, the repository keeper evaluates the new code updates to decide whether or not they are ready for release. In the absence of pull requests, incomplete or improperly written code modifications might be hastily merged into the main repository and might damage or interfere with the functioning of the live product. Pull requests only merge code that has been properly evaluated and authorized, preserving the product's integrity and enhancing the user experience.

Question 10: What is a “commit” in git?

The fundamental units of a Git project timeline are commits. Commits can be viewed as snapshots or checkpoints along a Git project's history. The git commit command creates commits to record the state of a project at that particular moment.

Question 11: Briefly explain the concept of GitHub Actions. What are they? How are they used? (/3 marks)

GitHub Actions is CI/CD platform that enables the user to automate build, test and deploy pipeline. The user may design workflows that deploy merged pull requests to production or build and test each pull request before adding it to his/her repository.
When an event takes place in the user's repository, such as the opening of a pull request or the creation of an issue, the user can set up a GitHub Actions workflow to be triggered. One or more jobs in the user's workflow are capable of running simultaneously or sequentially. Each job contains one or more steps that execute a script that you write or an action, a reusable extension that can streamline your workflow. Each job runs inside its own virtual machine runner or inside a container.

Question 12: What is a CI/CD pipeline? (/2 marks)

Continuous integration, continuous delivery, and continuous deployment are often referred to as CI/CD or CICD. In order to release a new version of software, a continuous integration and continuous deployment (CI/CD) pipeline is necessary. Through automation, CI/CD pipelines aim to enhance software delivery throughout the whole software development life cycle.