1. Create a git repository (create a directory and run **git init** in it). Create a text file with some content, and call it file.txt. Add the file to the index using **git add**, and commit it using **git commit**.

- What is the hash of the commit? Find the commit in the commit log using **git log**.
- Inspect the commit directly using **git cat-file -p <commit hash>**. What is the hash of the commit's tree?
- Inspect the tree using **git cat-file -p <tree hash>**. Which blobs does it contain, and what are their hashes?
- Inspect the content of the blobs using **git cat-file -p <blob hash>**. What is the content of the blobs?

2. Alice creates a file and checks it in on her computer. Bob creates a file which has the exact same content and file name, and checks it in with a commit message completely identical to Alice's. Are the hashes of their commits the same? Why or why not?

3. Alice creates a file and checks it in on her computer. Later that day, Alice creates a file which has the exact same content and file name, and checks it in with a commit message completely identical to her previous message. Are the hashes of their commits the same? Why or why not?

4. In the repository you created in (1), create a new branch with name "feature" on the initial commit and check it out using **git checkout -b feature**. Change the text file. Add the changed file to the index. Commit the changes. What is the hash of the new commit?

5. Check out the master branch again using **git checkout master**. Then run **git reflog** to inspect the log of which commits you have checked out. Can you find the first and second commits in this list?

6. Delete the feature branch using **git branch -D feature**. What does the commit graph look like now?

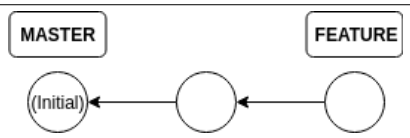
7. Let us now pretend that what you did in (6) was done by a mistake. You did not want to delete that feature branch Find the hash of the previous head of feature using git reflog. Then check the commit out directly using **git checkout <commit hash>**. You are now in a detached HEAD state. What does that mean? Recreate the feature branch on this commit using **git checkout -b feature**. What does the commit graph look like now?

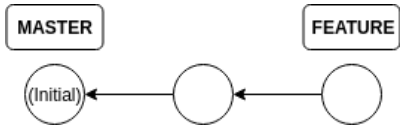
8. Which of the following merges are fast-forwardable?

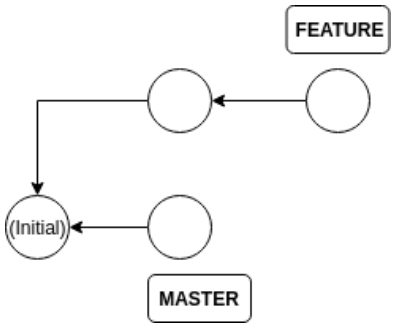
Remember: If you have checked out branch X and you run `git merge Y`, we say that you are "merging branch Y into branch X".

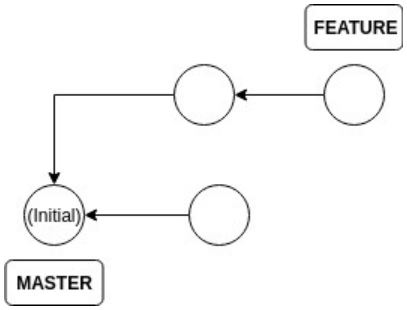
Remember: If you are merging branch Y into branch X, the merge is fast-forwardable if the head of branch X is on branch Y. In other words: if the head of branch X is an ancestor of the head of branch Y.

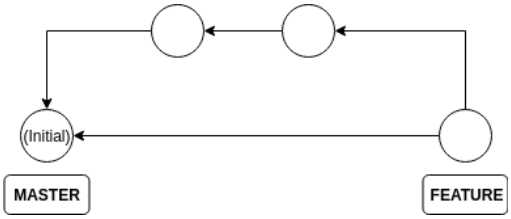
In the following diagrams, circles are commits, and square labels are branches.

A		Merge feature into master
		`git checkout master`
		`git merge feature`

B		Merge master into feature
		`git checkout feature`
		`git merge master`

C		Merge feature into master
		`git checkout master`
		`git merge feature`

D		Merge feature into master
		`git checkout master`
		`git merge feature`

E		Merge feature into master
		`git checkout master`
		`git merge feature`

9. In each of the cases above, draw the commit graph after the merge is done.