Assignment 1 (git intro)

1.1: Your first git repository

Install git (see `https://git-scm.com/`). Clone your private assignment repository. This repository should have the form UiO-INF3331/INF3331-X, where X is replaced by your UiO username. First, create a `assignment1` folder in your git repository. Then add a textfile in that directory, add it to the repository and commit it. The textfile should be named Readme.txt and contain the words "Hello world". Push your first commit to github.

Name of file: Readme.txt
Points: 3

1.2: Tagging

Tagging lets you label a commit with supplementary information. For example, when developing a program, you might tag the commit of the version you release as version 1.2 with the tag "v1.2". Use `git tag` to tag the commit you submit as this week's assignment with "week1_submission". Note that by default, git does not push tags for you, so you need to use `git push origin <TAGNAME>`.
Points: 2

1.2: Getting back old versions of files (optional)

This problem will teach you how to get old versions of a file. This is useful if you make a change you later end up regretting, or for debugging purposes. Add, commit and push a new file called `greeting.txt` containing a friendly greeting to your repository. Then change the file so the greeting is less friendly, and add, commit and push the modified file.

To fetch the old version, first use `git log` to get a list of the commits, and identify the commit containing the old version of the greeting, and note its commit hash (the string of letters/numbers after "commit "). Then use

```
git checkout COMMITHASH greeting.txt
```

where `COMMITHASH` is the commit hash you found using `git log`. This will replace the current version of `greeting.txt` with the version from the commit you chose, and `git add` it for you. Finally, use `git commit` to make a commit which reverts the greeting to the old version.

1.4: Resolve git conflicts (optional)

Pulling from a git repository fails if collaborators pushed conflicting changes since the last pull. In this case the merge of conflicting changes need to be performed manually. Try to simulate such a scenario and fix the resulting merge conflict:

1. Add, and push a new file `gitconflict.txt` to your git repository.
2. Clone your git repository again into a different directory.
3. In both repository directories, make different changes to `gitconflict.txt` and commit them separately.
4. Attempt to push the changes of both repositories to github. The second push will fail and you will need to resolve the conflict manually.