Assignment 3

This third assignment has multiple purposes.

- You will continue learning how to write functions using pattern matches, as well as recursive functions.
- You will continue practicing with version control.
- You will start managing your project by creating issues and milestones in the GitLab interface.

At this point you should have the following setup:

1. You created a fork of my functional−programming−assignments repository in GitLab. It probably has a web address that looks like https://gitlab.com/skiadas/functional−programming−as but with your name instead of mine.
2. You have also cloned your fork locally on your computers, and it resides in some folder probably called functional−programming−assignments.
   - You can check which remote you are connected to if you do the following from the command line from the functional−programming−assignments folder in your computer:

     ```
     git remote −v
     ```

     This should print two links, both associated with the remote called origin, one for push and one for fetch.
3. You have created a number of commits to your repository, first locally and then you pushed them to your fork.
4. In the meantime, I have added the next homework assignment to my repository. Since you made your fork before I did that, your fork does not have these changes. The first thing we will need to do is move those changes over to your fork, without overwriting the code you have committed. We will do this via the following steps:
   - You will first add my repository as a second "remote" to your local clone. We will call it upstream to distinguish it from the origin remote that is your fork. You only have to do this once for the course, while the following steps will need to be done each time you want to update your fork and clone with my "upstream" changes (e.g. when a new assignment comes out). You would do this with the following command (do NOT change the account name, use mine): git remote add upstream https://gitlab.com/skiadas/functional−programming−assignments.git If you were to do git remote −v now you should be seeing two more remote entries for upstream.

- We will do the remaining steps in GitKraken. Open your project in GitKraken and you should be seeing two remotes on the side, origin and upstream. In this step you will update the remote repositories to fetch the upstream update, and update your local upstream with the changes that happened on my repository. You do this with the little arrow to the right of the "Pull" menu, and you select the "Fetch All" option. You should also be seeing in the main window multiple paths. There is your local master, with a little laptop image to it, and if you have pushed recently then you would also see your fork "origin/master", with your gitlab avatar, at the same position. And you will probably see another branch, the "upstream/master", with my picture on it.
- Make sure the current branch is your local master, it should have a check-box next to it. Also make sure you have no uncommitted changes (they are usually in a WIP item at the top). Then right-click my branch's latests commit, and you should see a context menu. One of the options would say: "Merge upstream/master into master". Click it.
- You are ready to go! There should now be a folder containing the assignment3 descriptions. The README in that folder should tell you what you need to do to finish the assignment. BUT keep reading here, as you are expected to also create issues, and the following paragraphs describe that process.

Project Management

While this is is a relatively small project, we will ask you to use the various project management tools that GitLab and similar website provide, to organize your work. This may feel tedious at times, but it pays off on a larger-scale project. So getting used to the workflow is important.

We will discuss four components of project management in this section. You can find these components in the Issues tab on the left bar of your Gitlab project.

Issues Issues help identify individual tasks that need to be completed, and contain all relevant information regarding those tasks. Issues contain a number and a *title*, and an optional description. They can also contain *comments* added by you or other contributors to your project, so they can be a place for a conversation about the project, that is recorded for the future. You can also add *checklists* within the issue's comment, of individual subparts of the task that you can check off as you complete. Issues can be *open* or *closed* depending on whether they have been resolved or not. Looking at the closed issues shows you what work you have done already in your project, and looking at your open issues shows you what you still need to do.

You can link a commit to specific issue if you add a specific phrase to your commit summary. Ref #123 for example will link the commit to issue number 123, and a link to the commit will appear as a comment in the issue's page. The phrase Close #123 will go further and close the issue when you push.

Issues can also be assigned to individual team members. As you work on your assignment, you should create issues for tasks that need to be done or problems that need to be addressed, then assign the issues to yourself and add a suitable phrase to any commit that relates to this issue. You should NOT work on anything in your code unless there is a corresponding issue that you are trying to address.

Labels You can decorate an issue with any number of *labels*, that identify the kind of problem described by the issue. So you could have labels like "bug", "enhancement", "duplicate", "obsolete", "ui" and so on. You can create your own labels or let the system generate some default labels for you.

When looking at your issues, you can add labels to them, or you can also filter the issue list to only show you those issues with a particular label.

As you work on your assignment and create issues, you should assign *at least one* label to each issue. How you arrange those is entirely up to you.

Milestones Milestones allow you to set more long-term goals. A milestone has a title, a description, and start and due dates. Each issue can be added to exactly one milestone. Milestones can be closed when all their issues are completed.

As you work on each assignment, you should at the very least create one milestone for the assignment (e.g. the "Assignment 3" milestone). All issues pertaining to that assignment should be added to that milestone.

Boards Boards are a bit similar to milestones, but instead contain "lists" of issues. You can use them to set up an order of issues, and to manage which issue you are working on at any given time. Use of the board is optional, but I encourage you to familiarize yourself with it. At the very least having a list of "ToDo" and "Doing" issues would help.

Spend some time at the beginning of the assignment, as you read through the instructions, to create issues. For instance each function you need to write could be an issue. Though you have a certain degree of freedom regarding what issues to create and how many, you should at the very least create some issues.