Assignment 4

Thunks and symbol tables.

In this assignment we are going to work on two interesting topics. The one is thunks.The other is symbol tables. You will also learn to use the Milestones and Labels features in GitHub.

A *thunk* is essentially an unevaluated expression, which can be represented in OCAML as a function of type `unit -> 'a`, and as the value `fun () -> e`. This assignment asks you to construct various functions that operate on thunks (and so the inputs and return values of these functions are themselves functions).

Symbol tables are essentially tables on which you can store a *value* associated to a *key*. In our case the keys will be strings, and we will essentially be storing key-value pairs in a list that makes sure the keys are in increasing order.

In this assignment, you should NOT specify the types of your functions, but rather let the system figure it all out. This may make the resulting types appear different than what is required, you should make sure that they are indeed the same and that the only difference is because of type aliases.

As in prior assignments, you should not use any functions we have not learned about. These assignments are not about learning a number of library functions, but about delving deeply into fundamental building blocks of programming.

Correctness of the solutions, including the types and whether the code loads, will count for 15 points. Your code MUST load with no errors and have the correct types. Otherwise you will receive 0 for correctness even if most of your functions are correct.

Another 5 points will come from style issues, your tests and use of GitHub, including Milestones. You should create a GitHub issue for each "atomic" task that you have to deal with. For instance, creating tests for a specific function, implementing a specific function, polishing up a specific function, identifying an incorrect behavior in a function and writing a test for it, then fixing it, are all distinct issues.

Milestones are a feature of GitHub's Issues that effectively allow you to group issues together and set target dates for them. Item 2 below tells you how to create and manage milestones.

Labels are another feature of GitHub's Issues that allows you to attach one or more, well, labels to an issue to identify the kind of issue that it is. There are some default labels, and in item 3 below you will learn how to create your own labels and assign them to issues.

1. You already have created a link to the instructor's remote repository. You need to download the updated version of the instructor's repository and merge the changes into yours. The following should do that:

    - Type: `git fetch instr`

- Type: `git merge master instr/master`. This may give you a editor window to edit a commit message. You don't need to edit it, just "save" (writeOut) and exit/close the document. (This is probably not your prefered editor. See this page[1] for how to set up your favorite editor for use in these merge situations. I would recommend setting it to SublimeText).
- If it reports no problems, you're ready.
- If it reports merge conflicts, you will need to resolve those first, then make a commit. this page[2] has some instructions on how to do that. Or contact me.

2. In the GitHub issues page you should see two options, Labels and Milestones. Click on the Milestones option and create two milestones, one for Thunks and one for Symbol Tables. You have to choose a "due date" for them. When you later create an issue, you will see an option on the right side to assign that issue to a milestone. You should do this for all your issues on this assignment.

3. On the GitHub issues page there is a Labels option. Click on it to see the currently available labels and to create new ones. Create 3 new labels, one called "tests", another called "implementation" and a third called "cleanup". You can choose whatever colors you want. Also take a look at the existing labels. When you create a new issue, assign at least one of the three newly created labels to it. You can assign more labels if you think they are appropriate.

4. Throughout your work, create and close issues as described in the instructions above and in previous homeworks. Make sure to assign each issues to the correct milestone and to give them the correct labels.

5. You will find two files in this directory:

   - `assignment4sub.ml` This is the main submission file. It is where you will add your code for the various functions that you need to write. There are comments in that file to show you where to add your function definitions, and to tell you what your functions should do.
   - `assignment4tests.ml` This is a file with a small number of tests, and you should add plenty tests your own. "Tests" are arranged as lines `let ... = e` where `e` is a an expression that is meant to evaluate to a boolean indicating if the tests succeeded or not.

6. To "run" your tests, start an OCAML session in the terminal via `utop`, do:

```
#use "assignment4sub.ml";;
#use "assignment4tests.ml";;
```

[1] https://help.github.com/articles/associating-text-editors-with-git/
[2] https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/