Assignment 8

Building an Interpreter

In this assignment we will start work on building an interpreter for a programming language. You should have read the describing the interpreter[1] notes first.

Instead of instructions in the file itself, this time the instructions are provided in a Markdown document: https://github.com/skiadas/ProgLangAssignments/blob/master/assignment8doc.md

As before, 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. In some cases the system might derive a "more general type" than the expected one, and that is OK.

You will not need to write many functions from scratch. Most of this assignment involves adding components to existing functions and type declarations. This assignment is also unusual as it requires editing many different files at once.

As in prior assignments, you should not use any functions we have not learned about unless explicitly told to. These assignments are not about learning a number of library functions, but about delving deeply into fundamental building blocks of programming. You may use anything present in the Pervasives[2] or List[3] modules, though you really will not need much.

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. You should make sure that the compile steps described in the README in the assignment file work out without problems.

Another 5 points will come from style issues, your tests and use of GitHub, including Milestones and Labels, issues and closing those issues via commits. Keep creating issues for the various parts of the assignment. assigining them to milestones and tagging them with labels. **Create new milestones for each assignment**.

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[4] 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.

[1] ../notes/interpreter.html
[2] http://caml.inria.fr/pub/docs/manual-ocaml/libref/Pervasives.html
[3] http://caml.inria.fr/pub/docs/manual-ocaml/libref/List.html
[4] https://help.github.com/articles/associating-text-editors-with-git/

- If it reports merge conflicts, you will need to resolve those first, then make a commit. this page[5] has some instructions on how to do that. Or contact me.

2. Create new milestones for components of this assignment as described in the assignment's documentation file.
3. Throughout your work, create and close issues via commits as described in the instructions above and in previous homeworks. Make sure to assign each issue to the correct milestone and to give them the correct labels.
4. The assignment interacts with the files in the directory called interpreter. More details can be found in the assignment's documentation file.
5. See the README file in the interpreter folder for how to create and run tests.
6. The compile process produces many extra files. These are all automatically generated files and should not be committed. git should already be set up to ignore them, so this is not something you will need to worry about.

[5] https://help.github.com/articles/resolving-a-merge-conflict-from-the-command-line/