Assignment 3

Responsible lecturer: Yaron Gonen
Responsible TA: Noam Siegel
Submission Date: 10/5/2021

Submission Instructions

You are provided with the templates **ex3.zip**. Unpack the template files and open the folder named **ex3**. This is the project root. From the command line in that folder, invoke `npm install`, and work on the files in that directory, preferably working in the Visual Studio Code IDE (refer to the Useful Links). To run the tests, run `npm test` from the command line. Important: Do not add any extra libraries and do not change the provided package.json and tsconfig.json configuration files. The graders will use the exact provided files. If you find any missing necessary libraries, please let us know.

Save your answers to the theoretical questions (Part I) in a pdf file called **id1_id2.pdf** and save it in the project root. Complete the code for the programming questions (Part II) inside the missing sections of code in the **src** folder.

ZIP together your answers (including the pdf file, the src folder, and only these files) into a file called **id1_id2.zip**.

Further notes:

- Do not send assignment-related questions by e-mail, use the forum instead.
- Please open a request ticket in the Student Requests system for any administrative issues (milu'im/extensions/etc).
- Have Fun!!!

Part I: Theoretical Questions [35 points]

1. Is `let` in L3 a special form? Justify your answer.
2. In L3, what is the role of the function `valueToLitExp`?
3. The `valueToLitExp` function is not needed in the normal evaluation strategy interpreter (L3-normal.ts). Why?
4. The `valueToLitExp` function is not needed in the environment-model interpreter. Why?
5. What are the reasons that would justify switching from applicative order to normal order evaluation? Give an example.
6. What are the reasons that would justify switching from normal order to applicative order evaluation? Give an example.
7. In general, and as seen in class, substitution requires renaming. However, when the term that is substituted is "closed" (i.e., it does not contain free variables) then no renaming is required and naive substitution is correct.
    a. Prove it.
    b. Write evaluation rules for naive substitution
8. Draw an environment diagram for the following computation. Make sure to include the lexical block markers, the control links and the returned values.

```
(define a 2)
(define goo
 (lambda (x)
   (lambda (y)
     (/ x y))))
(define foo
 (letrec
     ((f (goo a))
      (g (lambda (x) (f x))))
   (lambda (x)
     (if (= x 0)
         x
         (g x)))))
(foo (foo 0))
```

Part II: Value Store [65 points]

In this part, you are given most of the implementation of the programming language *L2.1*. This language has almost the same syntax as the L2 language taught in class. The main difference is in the choice of evaluation model: it implements the Environment Model (L2 was implemented using the Substitution Model).
You are asked to change the implementation of the environment of L2.1 with two main features:

1. Add a set! expression support
2. Implement the mutation using a *Value Store*.

The Value Store

The value store plays the part of the *heap* in the execution: it is allocated dynamically during the execution and holds the actual value. The environment will now contain only the names of the variables and addresses of the value in the value store. The store is a partial function from these addresses to the values. Mutations occur only at the store. To do that, the value store will be implemented as an array of Box<Value> type.

Impact on the Evaluation

Store API

Functions for managing the store need to be added: makeStore, extendStore, applyStore.

Allocating Memory

The store needs to be extended whenever memory needs to be allocated. You need to figure out where, during the evaluation, this should be done.

Applying Environment

Since the environment holds only pointers to the store, and the actual values reside in the store, the process of fetching the values needs to be changed.

What to do

- Add `set!` to the syntax
- Complete the Store api
- Think when the store needs to be extended, and modify the code accordingly
- Pay attention to the global environment and define expressions
- Pay attention to closures and the environment they hold
- Make sure all the tests run

בהצלחה!