Homework: VarLang

Learning Objectives:

1. Get familiar with the concepts of scope, free and bound variables and environment
2. Write programs in VarLang
3. Understand and extend VarLang interpreter

Instructions:

1. Total points: 52 pt
2. Early deadline: Sept 25 (Wed) 11:59 pm, Regular deadline Sept 27 (Fri) 11:59 pm. (you can continue working on the homework till TA starts to grade the homework)
3. Download hw3code.zip from Canvas
4. Set up the programming project following the instructions in the tutorial from hw2 (similar steps)
5. How to submit:
   - For questions 1–3, you can write your solutions in latex or word and then convert it to pdf; or you can submit a scanned document with legible handwritten solutions. Please provide the solutions in one pdf file.
   - For questions 4–6, please submit your solutions in one zip file with all the source code files (just zip the complete project's folder).
   - Submit the zip file and one pdf file to Canvas under Assignments, Homework 3

Questions:

1. (4 pt) Write Varlang programs:
   (a) (1 pt) Compute the temperature in Celsius given a temperature $f = 58$ in Fahrenheit
   (b) (3 pt) Write a VarLang program that evaluates to value 0. The program must include at least 2 let expressions and there must be a "hole in one of the scopes."(The definition of this kind is seen in section 3.3 of Rajan-PL book pg 48), For example:
   (let ((x 2) (y 4)) (let ((x 5) (z 10)) (let ((z 20)) (- (* x y) z))))
   $ 0
2. (4 pt) Compute the values of the following Varlang expressions. (Return an error if the values cannot be computed.) Please list the steps to show that you understand the scoping rules and semantics of the VarLang program.

   Hint: section 3.5 of Rajan-PL book

(a) (+ (let ((y 1)) y) (let ((z 2) (x y)) (+ x z)))
(b) (let ((x 3) (y 0)) (let ((y x)) y))

3. (4 pt) List free and bound variables for the following Varlang expressions:

   (a) (let ((c 5) (z a)) (+ c (let ((x c) (y b)) (- z (+ x y)))))
   (b) (let ((a b)) (let ((x y) (y 10)) (+ c y x (- g a))))

4. (8 pt) Extend the interpreter: Let expression is useful to define multiple variables at the same time. However, one might want to refer to the previous defined variables in the same let expression when defining later variables. For example, in the evaluation of `(let ((a 3) (b a) (c (+ a b))) c)`, we created three variables, a, b and c, where a=3, b=a, c=a+b. Currently the Varlang intepreter will report "No binding found for name: a" This problem asks you to modify existing let evaluator to eliminate such errors and support this behavior. In the above example, *b* will get the value 3, c will get the value 6.

```
$ (let ((a 3) (b a) (c (+ a b))) c)
6
```

5. (15 pt) Extend the interpreter to support global constants and understand the idea of initial environment in programming languages. Extend the VarLang interpreter such that:
   (1) you can initialize the environment using *define VarExp Number*, and,
   (2) any VarLang programs that use such predefined constants can be evaluated accordingly.

```
$ (define month 9)
$ (let ((x 1)) (+ x month))
10
```

6. (20 pt) Extend the interpreter: Security is a major concern for any system. To deal with this it becomes important that deallocated memory of some program does not contain the data which can be read by malicious programs thereby causing leak of information. To avoid such information leak due to environment storage, we can augment the Varlang language with a *lete* (encoded let) expression that encodes the value before storing in environment and *dec* expression that decodes it prior to using it. Extend the Varlang programming language to support these two expressions. Implement an encrypted let (lete for let encrypted), which is similar to let but it uses a key and a dec expression that is similar to VarExp. All values are stored by encrypting them with key, and read by decrypting them with key.

```
$ (lete 2 ((x 1)) x)
3
$ (lete 20 ((x 1)) (dec 20 x))
1
$ (lete 10 ((y 8)) y)
18
$ (lete 10 ((y 12)) (dec 10 y))
12
```