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: 60 pts
2. Early deadline: Feb 24 (Wed) 11:59 pm, Regular deadline Feb 26 (Fri) 11:59 pm. (you can continue working on the homework till TA starts to grade the homework)
3. Download hw2code.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) [Varlang programming, scoping] Write Varlang programs:
   (a) (1 pt) Compute the temperature in Kelvin given a temperature $f = 58$ in Fahrenheit.
   (b) (3 pt) Write a VarLang program that evaluates to value 34. 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 2)) (let ((x 27) (z 10)) (let ((z 20)) (- (* x y) z))))
   $ 34
2. (4 pt) [Varlang syntax and semantics, scoping] 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) (2 pt) (let ((a 0) (b 4)) (let ((a b)) (+ a b)))

(b) (2 pt) (* (let ((b 4)) b) (let ((c 11) (a b)) (- a c)))

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

   (a) (2 pt) (let ((c 5) (d a)) (let ((e c) (f b)) (* c (- (* e f)))))

   (b) (2 pt) (let ((a b)) (let ((x y) (y 10)) (+ x y z (- g a))))

4. (15 pt) [Varlang interpreter basic] Extend the interpreter: Notice from the semantics and from the implementation of the let expression that Varlang doesn't place any restriction on defining the same variable two or more times in the same let expression. In fact, a variable can be defined any number of times and only the rightmost definition would have effect in the body of the let expression. So the program:
   (let ((a 3) (a 4) (a 2) (a 342)) a) would give the answer 342.

   Extend the Varlang programming language to support an unique let expression which is essentially a variation of current let expression where all variable names defined in the same *letu* expression must be unique.

   $ (letu ((a 3) (b 4)) (+ a b))
   7
   $ (letu ((a 3) (a 4)) a)
   error
   $ (letu ((a 3) (a 4) (a 2) (a 342)) a)
   error

5. (8 pt) [Environment] 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
   $ (define F 96454.56)(define R 10973731.6)

6. (25 pt) [Varlang interpreter advanced] Extend the interpreter: Security is a major concern for any system. Therefore, it is important to ensure that there is no information leak and that the memory deallocated from a program does not contain data which can be read by malicious programs. To avoid such a breach due to environment storage, we want to augment the Varlang language with a *lete* (encoded let) expression, to encode a value before storing it in the environment, and a *dec* expression to decode it before using it. All values are stored by encrypting them with key, and read by decrypting them with key.

   In the *lete* expression, each variable definition will contain the variable name, followed by two numbers where the first number is the value, and the second number is the encrytion key. In the *dec* expression, the first number is the key for decription, followed by the variable name.

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.

```
$ (lete ((x 1 2)) x)
3
$ (lete ((x 1 a)) x)
error
$ (lete ((x 1 20)) (dec 20 x))
1
$ (lete ((x 1 20)) (dec b x))
error
$ (lete ((y 8 10) (x 1 2)) (+ x y))
21
$ (lete ((y 12 10) (x 1 2)) (+ (dec 10 y) (dec 2 x)))
13
```