Problem Set 4
Comp 301
Fall 2020
Week 6: 09.11.2020 - 13.11.2020

Please use the code boilerplate for `LET` language, which includes several tests in `tests.rkt` for you to see if your code is correct. Uncomment corresponding test and run `tests.rkt`, ig you see the success message, then it means that your code works properly.

Save your code and zip it as `ID_username.zip` with your ID and username (Example: `1234567_galtintas17.zip`), and submit this ZIP file. You are expected to submit by the end of PS, however, you have an additional 1 hour to submit after the PS. The solutions will be available on the course BlackBoard after Friday. **Read the questions carefully. Good luck!**

Problem 1. : Write an `extend-env*` function, which works identical to `extend-env` but can take many parameters. Below is an example:

```
> (extend-env
   'i (num-val 1)
   (extend-env
      'v (num-val 5)
      (extend-env
        'x (num-val 10)
        (empty-env))))
((i #(struct:num-val 1)) (v #(struct:num-val 5)) (x #(struct:num-val 10)))

> (extend-env* '(x v i) (list (num-val 10) (num-val 5) (num-val 1))
   (empty-env))
((i #(struct:num-val 1)) (v #(struct:num-val 5)) (x #(struct:num-val 10)))
```

Hint: You need to make changes in `environments.rkt` and you can of course use the pre-existing `extend-env` in your function.

Problem 2. [1]: Extend the `let` language so that a let declaration can declare an arbitrary number of variables, using the grammar:

$$Expression ::= \text{let } \{Identifier = Expression\}^* \text{ in } Expression$$

As in Scheme's let, each of the right-hand sides are evaluated in the current environment and the body is evaluated with each new variable bound to the value of its associated right-hand side. For example:

```
let x = 30 in
   let x = -(x,1)
       y = -(x,2)
   in -(x,y)
```

evaluates to 1.

Note: You need to implement `extend-env*` in Problem 1 to be able to implement this problem. Hint: Don't forget to add `extend-env*` to the provide statement, that is at the beginning of `environments.rkt`, to use it.

[1] EOPL p.74 Exercise 3.16