Problem Set 10
COMP301 Fall 2021
Week 12: 23.12.2021 - 26.12.2021

Instructions:

- Submit your answers to the Blackboard PS10 assignment until December 26th Sunday, at 23.59.
- Please use the code boilerplate, which includes several tests for you to see if your code is correct.
- Submit your code and PDF file to BlackBoard as a single zip file *yourIDno_ username.zip*. (Example: *123456_fbulgur17.zip*)

Problem 1: [1] Modify the implementation of `proc-exp` to allow multiple arguments. Also change the implementation of `call-exp` to accept multiple operands.

Hint: You have implemented multi-argument procs before. Now try to do it in IREF.

New grammar of proc-exp and call exp

$$\begin{aligned} \mathit{Expression} &::= \texttt{proc}(\{\mathit{Identifier}\}^{*(,)})\ \mathit{Expression} \\ &::= (\mathit{Expression}\{\mathit{Expression}\}^{*}) \end{aligned}$$

Example

```
let x = 5 in let y = 6 in
   let z = proc (a,b) -(a,b) in (z x y)
   => returns (num-val -1)
```

Recommended Changes:

- **Note:** Carefully read the instructions commented in the code as well. More detailed information is given in the comments.
- Change `proc` datatype and implement `extend-env*` ,which extends environment with multiple var val pairs at once, in `data-structures.rkt`
- Change `apply-env` to handle the `extend-env*` in `environments.rkt`
- Change the implementations of grammars of `proc-exp` and `call-exp` accordingly in `lang.rkt`
- Change the implementation of `proc-exp`, `call-exp` inside `value-of` and also change `apply-procedure` to evaluate multi-argument procedures in `interp.rkt`.

Problem 2: [2] Modify the `let` expression so that it extends the environment with multiple pairs at once.

New grammar of let-exp

$$\mathit{Expression} ::= \texttt{let}\{\mathit{Identifier} = \mathit{Expression}\}^{*(,)}\ \texttt{in}\ \mathit{Expression}$$

Example

```
let x = 5, y = 6 in -(x,y)
   => returns (num-val -1)
```

Recommended Changes:

- **Note:** Carefully read the instructions commented in the code as well. More detailed information is given in the comments.
- Change the implementation of grammar of `let-exp` accordingly in `lang.rkt`
- Change the implementation of `let-exp` in `interp.rkt`
- **Hint:** You can use previously implemented `extend-env*`

[1] EOPL Exercise 4.17

[2] EOPL Exercise 4.17

Problem 3: Extend the IREF with `let*-exp` which takes multiple var and val pairs similar to problem 2. However, environment will be extended in order so that the example below can work without any errors.

Grammar for let*-exp

$$\mathit{Expression} ::= \texttt{let*}\{\mathit{Identifier} = \mathit{Expression}\}^{*(,)}\ \texttt{in}\ \mathit{Expression}$$

Example

```
let* x = 5, y = -(x,-1) in y
   => returns (num-val 6)
```

Recommended Changes:

- **Note:** Carefully read the instructions commented in the code as well. More detailed information is given in the comments.
- Implement the grammar of `let*-exp` accordingly in `lang.rkt`
- Implement `let*-exp` in `interp.rkt`

Problem 4: [3] It is suggested in the EOPL book: the use of assignment to make a program more modular by allowing one procedure to communicate information to a distant procedure without requiring intermediate procedures to be aware of it. Very often such an assignment should only be temporary, lasting for the execution of a procedure call. Add to the language a facility for dynamic assignment (also called fluid binding) to accomplish this. Use the production:

Grammar for setdyn-exp

$$\mathit{Expression} ::= \texttt{setdynamic}\ \mathit{Identifier} = \mathit{Expression}\ \texttt{during}\ \mathit{Expression}$$

The effect of the setdynamic expression is to assign temporarily the value of exp1 to var, evaluate body, reassign var to its original value, and return the value of body. The variable var must already be bound. For example:

Example

```
let x = 11
   in let p = proc (y) -(y,x)
      in -(setdynamic x = 17 during (p 22), (p 13))
      => returns (num-val 3)
```

In this code,the value of x which is in p, is set to 17 in the call (p 22) but is reset to 11 in (p 13), so the value of the expression is 5 - 2 = 3.

Recommended Changes:

- **Note:** Carefully read the instructions commented in the code as well. More detailed information is given in the comments.
- Implement the grammar of `setdyn-exp` accordingly in `lang.rkt`
- Implement `setdyn-exp` in `interp.rkt`

[3] EOPL Exercise 4.21