CS 496 – The PROC-Language
Exercise Booklet 4

Exercise 1

Write a derivation to show that `let f = proc (x){ x -11 } in (f 77)` is a program in PROC.

Exercise 2

Write down the parse tree for the expression `let pred = proc(x){ x-1 } in (pred 5)`.

Exercise 3

What are the two types of results (expressed values or `exp_val`) of a program in the LET-Language? What is the new type of result (expressed value or `exp_val`) that a program in the PROC-Language can produce as a result of its evaluation?

Exercise 4

Writ an OCaml expression of each of the OCaml types below:

1. `expr`
2. `env`
3. `exp_val`

Exercise 5

Execute the following program using the interpreter for PROC and copy the output on paper.

```
proc (x) { x-11 }
```

Exercise 6

Write down the result of evaluating the following expression:

```
proc (x) { let y=2 in x }
```

Depict the full details of the closure including the environment. Use the tabular notation seen in class to depict the environment.

Exercise 7

Write down the result of evaluating the following expression:

```
let a=1
in proc (x) { x }
```

Depict the full details of the closure including the environment.

Exercise 8

Write down the result of evaluating the following expression:

```
let a=1
in let b=2
in proc (x) { x }
```

Exercise 9

Write down the result of evaluating the following expression:

```
proc (x) { proc (y) { x-y }}
```

Exercise 10

Write down the result of evaluating the following expression:

```
let a=1
in proc (x) { proc (y) { x-y }}
```

Exercise 11

Consider the following code in PROC

```
1 let x=2
2 in let y=proc (d) { x }
3 in let z=proc(d) { x }
4 in 3
```

Draw the environment used by the interpreter when it is about to evaluate line 4.

Exercise 12

Use the "higher-order" trick of self-application to implement the mutually recursive definitions of `even` and `odd` in PROC:

$$\begin{array}{rcl} even(0) & = & true \\ even(n) & = & odd(n-1) \\ \\ odd(0) & = & false \\ odd(n) & = & even(n-1) \end{array}$$