Final Exam

COSE212 Programming Languages, Fall 2015

Instructor: Hakjoo Oh

Problem 1 (10pts) Natural numbers are inductively defined as follows:

$$n \quad \rightarrow \quad \circ \mid S\, n$$

where $\circ$ denotes 0, $S\,\circ$ denotes 1, $S\,(S\,\circ)$ denotes 2, and so on.

1. Define a function

$$\mathsf{add} : n \times n \rightarrow n$$

that adds two natural numbers.

$$\mathsf{add}(n_1, n_2) = \begin{cases} n_2 & n_1 = \circ \\ S\ (\mathsf{add}(n_1', n_2)) & n_1 = S\ n_1' \end{cases}$$

2. Define a function

$$\mathsf{mul} : n \times n \rightarrow n$$

that multiplies two natural numbers.

$$\mathsf{mult}(n_1, n_2) = \begin{cases} \circ & n_1 = \circ \\ \mathsf{add}(n_2, (\mathsf{mul}(n_1', n_2))) & n_1 = S\ n_1' \end{cases}$$

Problem 2 (10pts) The common pattern of the functions that accumulate something over a list can be captured by the higher-order function `fold`:

```
let rec fold f l a =
  match l with
  | [] -> a
  | hd::tl -> f hd (fold f tl a)
```

Re-write the following functions using `fold`:

1.
```
let rec length l =
  match l with
  | [] -> 0
  | hd::tl -> 1 + length tl
```

```
let length l = fold (fun e a -> 1 + a) l 0
```

2.
```
let rec append x y =
  match x with
  | [] -> y
  | hd::tl -> hd::(append tl y)
```

```
let append x y = fold (fun e a -> e::a) x y
```

Problem 3 (10pts) Consider the minimal yet Turing-complete programming language:

$$E \quad \rightarrow \quad x \mid \texttt{proc}\ x\ E \mid E\ E$$

1. Define its semantics with static scoping. The domain is given below.

$$\begin{array}{rcl} \mathit{Val} & = & \mathit{Procedure} \\ \mathit{Procedure} & = & \mathit{Var} \times E \times \mathit{Env} \\ \mathit{Env} & = & \mathit{Var} \rightarrow \mathit{Val} \end{array}$$

$$\frac{}{\rho \vdash x \Rightarrow \rho(x)}$$

$$\frac{}{\rho \vdash \texttt{proc}\ x\ E \Rightarrow (x, E, \rho)}$$

$$\frac{\rho \vdash E_1 \vdash (x, E, \rho') \quad \rho \vdash E_2 \Rightarrow v \quad \rho'[x \mapsto v] \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

2. Define its semantics with dynamic scoping. The domain is given below.

$$\begin{array}{rcl} \mathit{Val} & = & \mathit{Procedure} \\ \mathit{Procedure} & = & \mathit{Var} \times E \\ \mathit{Env} & = & \mathit{Var} \rightarrow \mathit{Val} \end{array}$$

$$\frac{}{\rho \vdash x \Rightarrow \rho(x)}$$

$$\frac{}{\rho \vdash \texttt{proc}\ x\ E \Rightarrow (x, E)}$$

$$\frac{\rho \vdash E_1 \vdash (x, E) \qquad \rho \vdash E_2 \Rightarrow v \qquad \rho[x \mapsto v] \vdash E \Rightarrow v'}{\rho \vdash E_1\ E_2 \Rightarrow v'}$$

Problem 4 (10pts) Convert the following programs into the lexical-address-based nameless representation:

1. `let a = 1 in let b = 2 in a + b`

```
let 1 in let 2 in #1 + #2
```

2.
```
let x = 3
  in proc (y)
     let z = (y - x)
     in (x - z + y)
```

```
let 3
 in proc
  let #0 - #1
  in #2 - #0 + #1
```

Problem 5 (10pts) Assuming static scoping for procedures, compare the behaviors and final values of the following two programs.

1.
```
let f = let counter = ref 0
        in proc (x) (counter := !counter + 1;
                     !counter)
  in let a = (f 0)
     in let b = (f 0)
  in (a - b)
```
-1

2.
```
let f = proc (x) (let counter = ref 0
                  in (counter := !counter + 1;
                      !counter))
  in let a = (f 0)
     in let b = (f 0)
  in (a - b)
```
0

Problem 6 (10pts) Infer the type of $(\lambda x.x)\ 1$:

$$\underbrace{\underbrace{(\lambda x.\underbrace{x}_{1})}_{2}\ \underbrace{1}_{3}}_{4}$$

1. (5pts) Generate type equations.
2. (10pts) Solve the equations using the unification algorithm. Explain each step clearly.

Problem 7 (20pts) Consider the following language:

$$E \rightarrow \mathsf{true} \mid \mathsf{false} \mid n \mid E_1 + E_2 \mid \text{if } E_1\ E_2\ E_3$$

and the lambda calculus:

$$L \rightarrow x \mid \lambda x.L \mid L_1\ L_2$$

We write $\underline{E}$ for the equivalent lambda term in L: that is, if E goes to a value v and $\underline{E}$ goes to a value l in lambda term, then $\underline{v} = l$. Define $\underline{E}$:

$$\begin{array}{rcl}
\underline{\mathsf{true}} & = & \lambda x.\lambda y.x \\
\underline{\mathsf{false}} & = & \lambda x.\lambda y.y \\
\underline{n} & = & \lambda s.\lambda z.s^n\ n \\
\underline{E_1 + E_2} & = & (\lambda n.\lambda m.\lambda s.\lambda z.m\ s\ (n\ s\ z))\ \underline{E_1}\ \underline{E_2} \\
\underline{\text{if } E_1\ E_2\ E_3} & = & \underline{E_1}\ \underline{E_2}\ \underline{E_3}
\end{array}$$

Problem 8 (20pts) O/X questions:

1. $\{3n \mid n \in N\}$ ($N = \{0, 1, 2, 3, \ldots\}$) is the only set S that satisfies the following two properties:

   (a) $0 \in S$, and

   (b) if $n \in S$, then $n + 3 \in S$

   X

2. Determining the values of program variables is a static property. X

3. C supports call-by-reference for procedure calls. X

4. Computers came first than programming languages. X

5. C's pointers, structs, set-jumps/long-jumps, gotos, local blocks, and loops are all syntactic sugars of eager-evaluating λ-calculus. O

6. All syntactically correct programs run OK in this language:

$$\begin{array}{rcl} C & \rightarrow & x := E \mid C; C \\ E & \rightarrow & Z \mid B \\ Z & \rightarrow & n \mid Z + Z \mid x \\ B & \rightarrow & \mathit{true} \mid \mathit{false} \mid Z < Z \end{array}$$

   X

7. There is only one redex in $((\lambda x.\lambda y.x)\ 1)\ 2$. O

8. The factorial function can be defined by

$$\mathit{fact} = Y(\lambda f.\lambda n.\text{if } n = 0 \text{ then } 1 \text{ else } n * f(n-1))$$

   where Y is the Y-combinator. O

9. We can design a sound and complete type system for Java. X

10. It is possible for the lambda calculus to simulate all language constructs of Java. O