Final Exam
COSE212 Programming Languages, Fall 2017

Name:
Student Number:

Write down answers only. Make sure that the answers are readable.

Problem 1 (15pts)

1. (5pts) Let S be the set of *all* propositional formulas defined over boolean constants ($true$ and $false$), negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), and implication ($\Longrightarrow$). Provide a bottom-up, inductive definition of S.

2. (5pts) Define S by rules of inference.

3. (5pts) Let P be the set of all palindromes over the binary alphabet $\Sigma = \{0, 1\}$. A string $x \in \Sigma$ is a palindrome if it reads the same forward and backward, e.g., ϵ (empty string), 0, 1, 00, 11, 010, 101, 0110, ... Define P by rules of inference.

Problem 2 (15pts)

1. (5pts) Write a higher-order function

   ```
   all : ('a -> bool) -> 'a list -> bool
   ```

   which decides if all elements of a list satisfy a predicate. For example, `all (fun x -> x mod 2 = 0) [1;2;3]` evaluates to false while `all (fun x -> x > 5) [7;8;9]` is true.

   ```
   let rec all p l =
   match l with
   | [] -> (1)
   | hd::tl -> (2)
   ```

   Complete (1) and (2).

2. (5pts) Write a higher-order function

   ```
   dropWhile : ('a -> bool) -> 'a list -> 'a list
   ```

   which removes elements of a list while they satisfy a predicate. E.g., `dropWhile (fun x -> x mod 2 = 0) [2;4;7;9]` is `[7;9]` and `dropWhile (fun x-> x>5) [1;3;7]` is `[1;3;7]`.

   ```
   let rec dropWhile p l =
   match l with
   | [] -> (1)
   | hd::tl -> (2)
   ```

   Complete (1) and (2).

3. (5pts) Let us define the type of natural numbers as follows:

   ```
   type nat = Zero | Succ of nat
   ```

   Define a function `nat2int : nat -> int`, which converts natural numbers to integers. For example, `nat2int (Succ (Succ (Succ Zero)))` evaluates to 3.

   ```
   let rec nat2int n =
   match n with
   | Zero -> (1)
   | Succ m -> (2)
   ```

   Complete (1) and (2).

Problem 3 (10pts) JavaScript programs often use closures: e.g.,

```
function create(init) {
  return function (step) {
    init += step;
    return init;
  };
}
var inc = create(5);
var a = inc(1);
var b = inc(2);
```

What are the values of `a` and `b` at the end of the program? Hint: the JavaScript code can be translated to our language with implicit references as follows:

```
let create =
  proc (init)
    (proc (step) (set init = init + step; init)) in
let inc = (create 5) in
let a = (inc 1) in
let b = (inc 2) in
(* What are the values of a and b here? *)
```

Problem 4 (10pts) What is the value of the program?

```
let a = 1 in
  let b = 2 in
    let p = proc (a) (a+b) in
      let f = proc (b) (p (b+1)) in
        let a = 5 in
          (f 2)
```

1. (5pts) With static scoping:
2. (5pts) With dynamic scoping:

Problem 5 (10pts) Is lazy evaluation always faster than eager evaluation? If yes, explain why. If not, illustrate a counter-example.

Problem 6 (10pts) Consider the programs:

```
(a)
let x = read in
  letrec double(x) =
    if iszero x then 0 else (double (x-1)) + 2 in
      double

(b)
letrec fact(n) =
  if iszero n then 1 else ((fact (n-1)) * n) in
    (fact read)

(c)
proc (maker)
  proc (x)
    if iszero (x) then 0
    else (((maker maker) (x-1)) + 4)

(d)
let f = proc (x) (x-11) in
  (f (f 77))

(e)
let f = proc (x) x in
  (f f)
```

1. (5pts) Choose all programs that are accepted by the simple (monomorphic) type system.
2. (5pts) Choose all programs that are accepted by the let-polymorphic type system.

Problem 7 (10pts) Let E be the simple expression language:

$$E \to n \mid \mathit{true} \mid \mathit{false} \mid E_1 + E_2 \mid E_1 < E_2 \mid E_1 \&\& E_2$$

where n denotes a natural number, true true, false false, $E_1 + E_2$ addition, $E_1 < E_2$ is true iff the number denoted by E_1 is less than E_2, $E_1 \&\& E_2$ is true iff both E_1 and E_2 are true.

1. (5pts) The meaning of an expression is the (integer or boolean) value denoted by the expression. Define the evaluation rules.
2. (5pts) Design a sound and complete type system for the language.

Problem 8 (20pts) O/X questions. 1 point each. Leave a blank when you are uncertain; each correct answer gets you 2 points but you lose 2 points for each wrong answer.

1. The set of even numbers, $\{0, 2, 4, \ldots\}$, can be defined as the smallest set $S \subseteq \mathbb{N}$ such that
   - $0 \in S$, and
   - $F(S) \subseteq S$ where $F(S) = \{n + 2 \mid n \in S\}$.
2. The set S defined above is unique.
3. In C, type-checking is done at compile-time.
4. C supports dynamic scoping.
5. C supports call-by-reference.
6. A type system that always accepts input programs is complete.
7. In a language with explicit references, locations are first-class objects.
8. Any Turing-complete language can be translated into lambda calculus.
9. Types in programming languages are a static property.
10. Recall the Church encoding of booleans: true and false are represented by $\lambda t.\lambda f.t$ and $\lambda t.\lambda f.f$, respectively. We can define the logical-and (i.e., &&) operator as follows:

$$\lambda b.\lambda c.b\ c\ \mathit{false}$$