Name:

Midterm 2

CMSC 430
Introduction to Compilers
Spring 2015

April 23, 2015

Instructions

This exam contains 7 pages, including this one. Make sure you have all the pages. Write your name on the top of this page before starting the exam.

Write your answers on the exam sheets. If you finish at least 15 minutes early, bring your exam to the front when you are finished; otherwise, wait until the end of the exam to turn it in. Please be as quiet as possible.

If you have a question, raise your hand. If you feel an exam question assumes something that is not written, write it down on your exam sheet. Barring some unforeseen error on the exam, however, you shouldn't need to do this at all, so be careful when making assumptions.

Question	Score	Max
1		15
2		15
3		25
Total		55

Question 1. Short Answer (15 points).

a. (5 points) Briefly explain the difference between type *inference* and type *checking*.

Answer: Type inference starts from a program without type annotations and tries to construct annotations that make the program well-typed.

Type checking starts from a program with type annotations and tries to confirm that the program is well-typed.

b. (5 points) In a few sentences, describe how a method invocation $o.m(a_1, \ldots, a_n)$ ("invoke method m on object o with arguments $a_1, \ldots, a_n$") could be carried out by a virtual machine.

Answer: The virtual machine accesses the object o's class name and retrieves the corresponding virtual table to see if it contains m. If so, it calls the method body passing o (as `this`) and arguments $a_1 \ldots a_n$. If not, it tries again starting from the object's superclass.

c. (5 points) Briefly explain the purpose of a *closure*.

Answer: The purpose of a closure is represent a delay substitution; it pairs together an expression to be evaluated later and an environment giving meanings to the free variables in the expression.

Question 2. Program transformations (15 points).

a. (15 points) Apply defunctionalization to this program:

```
type bt =
 | Leaf
 | Node of int * bt * bt

let rec sumk b k =
  match b with
  | Leaf → k 0
  | Node (i, b1, b2) →
     sumk b1 (fun sb1 → sumk b2 (fun sb2 → k (i + sb1 + sb2)))
let sum b =
  sumk b (fun sb → sb)
```

Answer:

```
type k =
 | K0
 | K1 of bt * int * k
 | K2 of int * int * k

let rec sumk b k =
  match b with
  | Leaf → apply k 0
  | Node (i, b1, b2) →
     sumk b1 (K1 (b2, i, k))
and apply k n =
  match k with
  | K0 → n
  | K1 (b2, i, k) →
     let sb1 = n in
     sumk b2 (K2 (i, sb1, k))
  | K2 (i, sb1, k) →
     let sb2 = n in
     apply k (i + sb1 + sb2)

let rec sum b =
  sumk b K0
```

Question 3. Type Systems (25 points).

a. (8 points) Assume that $int < float$. Write down every type t such that $t \leq float \to int \to float$, following standard subtyping rules.

Answer:

1. $float \to int \to float$
2. $float \to float \to float$
3. $float \to int \to int$
4. $float \to float \to int$

b. (2 points) Assume that $int < float$. Write down every type t such that $t \leq int\ ref \to float\ ref$, following standard subtyping rules.

Answer:

$$int\ ref \to float\ ref$$

c. (5 points) Recall the simply typed lambda calculus:

$$\begin{array}{rcl} e & ::= & n \mid x \mid \lambda x{:}\, t.e \mid e\ e \\ t & ::= & \mathit{int} \mid t \to t \\ A & ::= & \emptyset \mid x{:}\, t, A \end{array}$$

$$\textsc{Int}\ \frac{}{A \vdash n : \mathit{int}} \qquad \textsc{Var}\ \frac{}{A \vdash x : A(x)} \qquad \textsc{Lam}\ \frac{x{:}\, t, A \vdash e : t'}{A \vdash (\lambda x{:}\, t.e) : t \to t'} \qquad \textsc{App}\ \frac{A \vdash e_1 : t \to t' \qquad A \vdash e_2 : t}{A \vdash e_1\ e_2 : t'}$$

Draw a derivation that the following type judgment holds, where $A = +{:}\, \mathit{int} \to \mathit{int} \to \mathit{int}$. (You can draw the derivation upward from the judgment, and you can write i instead of int to save time):

Answer:

$$\nabla_1 = \frac{\dfrac{\overline{f{:}\, \mathit{int} \to \mathit{int}, A \vdash f : \mathit{int} \to \mathit{int}} \qquad \overline{f{:}\, \mathit{int} \to \mathit{int}, A \vdash 1 : \mathit{int}}}{f{:}\, \mathit{int} \to \mathit{int}, A \vdash f\ 1 : \mathit{int}}}{A \vdash \lambda f{:}\, \mathit{int} \to \mathit{int}.f\ 1 : (\mathit{int} \to \mathit{int}) \to \mathit{int}}$$

$$\nabla_2 = \frac{\dfrac{\dfrac{\overline{x{:}\, \mathit{int}, A \vdash + : \mathit{int} \to \mathit{int}} \qquad \overline{x{:}\, \mathit{int}, A \vdash 1 : \mathit{int}}}{x{:}\, \mathit{int}, A \vdash + \ 1 : \mathit{int} \to \mathit{int}} \qquad \overline{x{:}\, \mathit{int}, A \vdash x : \mathit{int}}}{x{:}\, \mathit{int}, A \vdash + \ 1\ x : \mathit{int}}}{A \vdash \lambda x{:}\, \mathit{int}.+\ \ 1\ x : \mathit{int}}$$

$$\frac{\nabla_1 \qquad \nabla_2}{A \vdash ((\lambda f{:}\, \mathit{int} \to \mathit{int}.f\ 1)\ (\lambda x{:}\, \mathit{int}.+\ \ 1\ x)) : \mathit{int}}$$

$$A \vdash ((\lambda f{:}\, \mathit{int} \to \mathit{int}.f\ 1)\ (\lambda x{:}\, \mathit{int}.+\ \ 1\ x)) : \mathit{int}$$

d. (10 points) Perform type inference on the following program by listing the types that OCaml will infer for the blanks:

```
let rec bumble (f : ___) (xs : ___) : ___ =
  match xs with
  | [] → []
  | x :: xs →
     ((x ("fred" ^ (f 0))) + 1) :: (bumble f xs)
```

Answer:

```
let rec bumble (f : (int → string)) (xs : (string → int) list) : int list =
  match xs with
  | [] → []
  | x :: xs →
     ((x ("fred" ^ (f 0))) + 1) :: (bumble f xs)
```