Name:

Midterm 2 (Practice Exam)

CMSC 430

Introduction to Compilers

Spring 2015

April 21, 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 what a *virtual method table* (or *vtable*) is and what it's used for.

Answer: It is a collection of methods for a particular class. Each instance of a class has a pointer to the virtual method table for that class. When one of an object's instance methods is invoked, it is resolved by looking it up in the virtual method table.

b. (5 points) Briefly describe the role of an *environment* in an interpreter or compiler.

Answer: An environment maps variables to their meaning (values). It is an efficient implementation of substitution.

c. (5 points) Briefly explain the goal of defunctionalization.

Answer: Defunctionalization is a program transformation that eliminates the need for function values in a program.

Question 2. Program transformations (15 points).

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

```
let rec fibk n k = match n with
  | 0 → k 0
  | 1 → k 1
  | n → fibk (n−1) (fun fn1 → fibk (n−2) (fun fn2 → k (fn1 + fn2))
let fib n = fibk n (fun n → n)
```

Answer:

```
type k = K0
       | K1 of int ∗ k
       | K2 of int ∗ k

let rec fibk n k = match n with
  | 0 → apply k 0
  | 1 → apply k 1
  | n → fibk (n−1) (K1 (n, k))
and apply k n = match k with
  | K0 → n
  | K1 (m, k) → fibk (m−2) (K2 (n, k))
  | K2 (m, k) → apply k (m+n)
let fib n = fibk n K0
```

Question 3. Type Systems (25 points).

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

Answer:

$$
\begin{array}{l}
int \rightarrow float \rightarrow int \\
int \rightarrow float \rightarrow float \\
float \rightarrow float \rightarrow int \\
float \rightarrow float \rightarrow float
\end{array}
$$

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

Answer:

$$int\ ref \rightarrow float\ ref$$

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

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

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

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

Answer:

$$
\cfrac{\cfrac{\cfrac{x\!: int, A \vdash + : int \to int \to int \qquad x\!: int, A \vdash x : int}{x\!: int, A \vdash +\ x : int \to int}}{A \vdash (\lambda x\!: int.\ +\ x) : int \to int \to int} \qquad \cfrac{}{A \vdash 1 : int}}{A \vdash (\lambda x\!: int.\ +\ x)\ 1 : int \to int}
$$

$$A \vdash (\lambda x\!: int.\ +\ x)\ 1 : int \to int$$

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

```
let rec mumble (f : ___) (xs : ___) (ys : ___) : ___ =
  match xs, ys with
  | [], [] -> []
  | x :: xs, y :: ys ->
      ((f (x+1))-1,(f (y-1))+1)::(mumble f xs ys)
```

Answer:

```
let rec mumble (f : (int -> int)) (xs : int list) (ys : int list) : (int * int) list = ...
```