Computer Science 2SC3 and Software Engineering 2S03

Final Exam Answer Key

DAY CLASS Dr. William M. Farmer
DURATION OF EXAMINATION: 3 Hours
MCMASTER UNIVERSITY FINAL EXAMINATION December 2008

THIS EXAMINATION PAPER INCLUDES 11 PAGES AND 34 QUESTIONS. YOU ARE RESPONSIBLE FOR ENSURING THAT YOUR COPY OF THE PAPER IS COMPLETE. BRING ANY DISCREPANCY TO THE ATTENTION OF YOUR INVIGILATOR.

Special Instructions

The use of any notes and books is permitted during this exam, but you may not use any calculators or other electronic devices. Answers to the first thirty questions (1–30) are to be marked on the OMR scan sheet. Answer the last four questions (31–34) in the space provided on the exam. Do **NOT** use correction fluid on the exam or on the OMR scan sheet. An answer key will be posted on the course web site. Good luck!

OMR Examination Instructions

NOTE: IT IS YOUR RESPONSIBILITY TO ENSURE THAT THE ANSWER SHEET IS PROPERLY COMPLETED: YOUR EXAMINATION RESULT DEPENDS UPON PROPER ATTENTION TO THESE INSTRUCTIONS.

The scanner, which reads the sheets, senses the shaded areas by their nonreflection of light. A heavy mark must be made, completely filling the circular bubble, with an HB pencil. Marks made with a pen or felt-tip marker will **NOT** be sensed. Erasures must be thorough or the scanner may still sense a mark. Do **NOT** use correction fluid on the scan sheet. Do **NOT** put any unnecessary marks or writing on the sheet.

(1) Print your name, student number, course name, section number, and the date in the space provided at the top of SIDE 1 (red side) of the form. The sheet **MUST** be signed in the space marked SIGNATURE.

(2) Mark your student number in the space provided on the sheet on SIDE 1 and **fill in the corresponding bubbles underneath**.

(3) Mark only **ONE** choice from the alternatives (A,B,C,D,E or 1,2,3,4,5) provided for each question. For a True/False question, enter a response of A or 1 for True and B or 2 for False. The question number is to the left of the bubbles. Make sure that the number of the question of the scan sheet is the same as the question number on the exam.

(4) Pay particular attention to the Marking Directions on the form.

(5) Begin answering questions using the first set of bubbles, marked "1".

Continued on page 2

(1) [2 pts.] Without requirements there is no basis for judging the correctness or quality of a program. Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

(2) [2 pts.] The lambda expression $\lambda x : \mathbf{R} \,.\, \lambda y : \mathbf{R} \,.\, x + y$ represents addition as a curryed function on the real numbers. Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

(3) [2 pts.] In C, recursion should be used when clarity is more important than efficiency. Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

(4) [2 pts.] From the point of view of efficiency, loops are a necessary part of the OCaml language. Is this statement true or false?

A.) True.

B.) $\boxed{\text{False.}}$

(5) [2 pts.] At the requirements level there is essentially no difference between a record of type t and a pointer to a record of type t. Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

(6) [2 pts.] The iterator function defined in Programming Exercise 5 is higher order but not polymorphic. Is this statement true or false?

A.) True.

B.) $\boxed{\text{False.}}$

(7) [2 pts.] Exceptions are good for implementing functions that are not defined on all members of the input type(s). Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

Continued on page 3

(8) [2 pts.] Programming languages usually have a precise syntax and semantics. Is this statement true or false?

A.) True.

B.) $\boxed{\text{False.}}$

(9) [2 pts.] In C, if `x` is a pointer, then `x` and `&x` return the same value. Is this statement true or false?

A.) True.

B.) $\boxed{\text{False.}}$

(10) [2 pts.] Unlike an OCaml for loop, a C for loop can simulate a while loop. Is this statement true or false?

A.) $\boxed{\text{True.}}$

B.) False.

(11) [2 pts.] Which numeric type in C corresponds to the `float` type in OCaml?

A.) `float`.

B.) `single`.

C.) $\boxed{\texttt{double.}}$

D.) None of the above.

(12) [2 pts.] Which of the following is an imperative statement in English?

A.) $\boxed{\text{Step aside.}}$

B.) Long live computing!

C.) A fisher is a big member of the weasel family.

D.) Who are you?

(13) [2 pts.] When the C preprocessor applies the macro

```
#define add(a,b) = a + b
```

to the expression `add(2 + 3,4)`, the result is the expression

A.) $\boxed{\texttt{2 + 3 + 4.}}$

B.) `5 + 4`.

C.) `9`.

D.) `add(add(2,3),4)`.

Continued on page 4

(14) [2 pts.] Which of the following declares in C a function pointer named `bongo`?

A.) `int bongo(int x, int y);`

B.) `int * bongo(int x, int y);`

C.) `int *bongo(int x, int y);`

D.) $\boxed{\texttt{int (*bongo)(int x, int y);}}$

(15) [2 pts.] A goto statement is a

A.) A sequential control structure.

B.) A conditional control structure.

C.) Iterative control structure.

D.) $\boxed{\text{None of the above.}}$

(16) [2 pts.] In OCaml, a variable of type `int ref` can be used in essentially the same way as a variable of type

A.) `int`.

B.) {`x : int`}.

C.) $\boxed{\{\texttt{mutable x : int}\}.}$

D.) `int array`.

(17) [2 pts.] The value of the following OCaml phrase is a function:

```
function goat -> sheep ;;
```

What is the name of the function.

A.) `function`.

B.) `goat`.

C.) `sheep`.

D.) $\boxed{\text{It has no name.}}$

(18) [2 pts.] Suppose we would like to implement in OCaml a list of records of type `rec`. If access speed is the primary concern, which of the following kinds of OCaml data structures would be the best choice for holding the list?

A.) List.

B.) $\boxed{\text{Array.}}$

C.) Linked list.

D.) Stack.

Continued on page 5

(19) [2 pts.] Which of the following expressions in C is not strictly evaluated by the call by value strategy.

A.) `(a == b) ?  c : d`

B.) `a && b`

C.) `a || b`

D.) $\boxed{\text{All of the above.}}$

(20) [2 pts.] Which of the following C functions is tail recursive?

A.) `int f(int x) { return f(x) + 1; }`

B.) `int f(int x) { return f(x-2) + f(x-1); }`

C.) `int f(int x) { return f(f(x)); }`

D.) $\boxed{\text{None of the above.}}$

(21) [2 pts.] Suppose the following phrase has been evaluated in OCaml;

```
let p = (1,2,3,4) ;;
```

Which expression evaluates to 3?

A.) `fst (snd (snd p))`.

B.) `hd (tl (tl p))`.

C.) $\boxed{\texttt{match p with (a,b,c,d) -> c.}}$

D.) All of the above.

(22) [2 pts.] Which of the following kinds of data structures is best for holding a piece of program code?

A.) Array.

B.) String.

C.) Linked list.

D.) $\boxed{\text{Tree.}}$

(23) [2 pts.] Which of the following kinds of data structures is best for holding a queue whose length will never be greater than 10?

A.) $\boxed{\text{Array.}}$

B.) Linked list.

C.) Stack.

D.) Tree.

Continued on page 6

(24) [2 pts.] Which of the following kinds of OCaml data structures is best for simultaneously holding a value of type t_1 and a value of type t_2?

A.) Sum.

B.) $\boxed{\text{Product.}}$

C.) List.

D.) Array.

(25) [2 pts.] Which of the following kinds of OCaml data structures is best for holding a value of type t_1 or t_2?

A.) $\boxed{\text{Sum.}}$

B.) Product.

C.) List.

D.) Array.

(26) [2 pts.] Let `p` be a pointer in C. Which of the following C expressions evaluates to true?

A.) `p + 7 == p[7]`.

B.) $\boxed{\texttt{p + 7 == \&p[7].}}$

C.) `*p + 7 == p[7]`.

D.) `*(p + 7) == &p[7]`.

(27) [2 pts.] According to the Principle of Least Privilege, which of the following is most desirable?

A.) Local variable.

B.) $\boxed{\text{Local constant.}}$

C.) Global variable.

D.) Global constant.

(28) [2 pts.] Which of the following OCaml expressions evaluates to a mutable data structure?

A.) `(x,y,z)`.

B.) `(x;y;z)`.

C.) `[x;y;z]`.

D.) $\boxed{\texttt{[|x;y;z|].}}$

Continued on page 7

(29) [2 pts.] The program `ocamlc`

A.) Interprets OCaml programs.

B.) $\boxed{\text{Compiles OCaml programs to bytecode.}}$

C.) Compiles OCaml programs to native machine code.

D.) Translates OCaml programs to C code.

(30) [2 pts.] Both of the C and OCaml programming languages include

A.) Macros.

B.) Garbage collection.

C.) $\boxed{\text{Recursively defined procedures.}}$

D.) Type inference.

Continued on page 8

(31) [10 pts.] Suppose the following OCaml phrases have been evaluated:

```
type 'a node_rec = {
  mutable data : float;
  mutable next: 'a
} ;;

type node =
    Null
  | Normal of node node_rec ;;
```

Using a while loop, write an OCaml function

```
length : node -> int
```

that, given a node n (of type `node`) as input, returns as output the number of `Normal` nodes in the linked list starting at n. You may assume that the linked list is not circular.

Answer:

```
let length n =
  let count = ref 0 in
  let node = ref n in
  while !node <> Null do
    match !node with
    | Null -> ()
    | Normal r ->
        begin
          count := !count + 1 ;
          node := r.next ;
        end
  done ;
  !count ;;
```

Continued on page 9

(32) [10 pts.] *Using recursion*, write an OCaml function

```
map : ('a -> 'b) -> 'a list -> 'b list
```

that, given a function f (of type `'a -> 'b`) and a list k (of type `'a list`) as input, returns as output a list k' (of type `'b list`) such that k' is obtained from k by applying f to each member of k. For example, `map (function x -> 0) [3;4;5]` evaluates to `[0,0,0]`.

Note: You will receive 2 bonus points if you (correctly) use pattern matching.

Answer:

```
let rec map f = function
  | [] -> []
  | head :: tail -> (f head) :: (map f tail) ;;
```

Continued on page 10

(33) [10 pts.] The *dot product* of two lists $[r_1, \ldots, r_n]$ and $[s_1, \ldots, s_n]$ of real numbers is

$$(r_1 * s_1) + \cdots + (r_n * s_n).$$

Using a for loop, write a C function with the prototype

```
double dot(double x[10], double y[10]);
```

that, given two arrays x and y of length 10 as input, returns as output the dot product of the two lists represented by x and y.

Answer:

```
double dot(double x[10], double y[10]) {
  double accum = 0;
  int i;
  for (i = 0; i < 10; i++)
    accum = accum + (x[i] * y[i]);
  return accum;
}
```

Continued on page 11

(34) [10 pts.] Assume the following type definitions have been made in C:

```
typedef struct n_rec {
  double data;
  struct n_rec * next;
  struct n_rec * previous;
} node_rec;

typedef node_rec * node;
```

Write a function in C with prototype

```
void insert_after(node n, double d);
```

that, given a node n and a datum d as input, creates a node n' holding d and inserts n' into the doubly linked list containing n after n (so that n' is the next node after n).

Answer:

```
void insert_after(node n, double d) {
  node new = malloc(sizeof(node_rec));
  new->data = d;
  new->next = n->next;
  new->previous = n;
  if (n->next != NULL)
    (n->next)->previous = new;
  n->next = new;
  return;
}
```

The End