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 2009

THIS EXAMINATION PAPER INCLUDES 12 PAGES AND 32 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 twenty-six questions (1–26) are to be marked on the OMR scan sheet. Answer the last six questions (27–32) 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.] Which programming language adheres the closest to the Principle of Least Privilege?

A.) [OCaml.]

B.) C.

(2) [2 pts.] Which programming language has the strongest separation between the language and its implementation?

A.) [OCaml.]

B.) C.

(3) [2 pts.] Which programming language is the most permissive?

A.) OCaml.

B.) [C.]

(4) [2 pts.] Which programming language makes the programmer responsible for memory management?

A.) OCaml.

B.) [C.]

(5) [2 pts.] In C, evaluation of the assignment statement

```
x = e;
```

always results in an error if the variable `x` and the expression `e` have different types. Is this statement true or false?

A.) True.

B.) [False.]

(6) [2 pts.] In both OCaml and C, it is possible to obtain the memory address of any data structure. Is this statement true or false?

A.) True.

B.) [False.]

(7) [2 pts.] A function in OCaml of type `unit -> unit` is useless. Is this statement true or false?

A.) True.

B.) [False.]

Continued on page 3

(8) [2 pts.] In OCaml, multi-ary functions can be easily represented by unary functions? Is this statement true or false?

A.) True.

B.) False.

(9) [2 pts.] The existence of an invariant for a loop proves that the loop is correct. Is this statement true or false?

A.) True.

B.) False.

(10) [2 pts.] In C, what kind of code file does not need a corresponding header file?

A.) One that implements a module.

B.) One that contains only functions.

C.) One that contains only macros.

D.) One that will never be used by any other code file.

(11) [2 pts.] Suppose that we want a function in C named `diomedes` to modify the values of three variables `a`, `b`, and `c` of type `int`. What should the function prototype and function application of `diomedes` look like?

A.) `void diomedes(int x, int y, int z);` and `diomedes(a,b,c)`.

B.) `void diomedes(ref int x, ref int y, ref int z);` and `diomedes(a,b,c)`.

C.) `void diomedes(int * x, int * y, int * z);` and `diomedes(&a,&b,&c)`.

D.) `void diomedes(int x, int y, int z);` and `diomedes(&a,&b,&c)`.

(12) [2 pts.] A variable in C of type `int *` corresponds to a variable in OCaml of type

A.) `int`.

B.) `int *`.

C.) `int ref`.

D.) `(int ref) ref`.

(13) [2 pts.] What is the syntax for a *block* in C?

A.) `(···)`.

B.) `[···]`.

C.) `{···}`.

D.) `begin···end`.

Continued on page 4

(14) [2 pts.] Let f be an OCaml function of type

```
'a * 'b -> 'c
```

and g be an OCaml function of type

```
'a -> 'b -> 'c.
```

If g is the curryed form of f, then

A.) $f = g$.

B.) $f\ x = g\ x$ for all values x.

C.) $f\ x\ y = g\ x\ y$ for all values x and y.

D.) $f\ (x, y) = g\ x\ y$ for all values x and y.

(15) [2 pts.] Which of the following definitions in OCaml defines a function by tail recursion?

A.) `let rec f x = if x = 0 then f(x - 1) + x else f(x - 2) ;;`

B.) `let rec g x = if x = 0 then g(x - 1) + g(x - 2) else 5 ;;`

C.) `let rec h x = h(h(x - 1)) ;;`

D.) `let rec i x = if x = 0 then i(x - 1) else i(x - 2) ;;`

(16) [2 pts.] OCaml, C#, and Java all

A.) Utilize garbage collection.

B.) Have compilers that translate source code to bytecode.

C.) Are object oriented.

D.) All of the above.

(17) [2 pts.] Which of the following types could be defined by an algebraic data type?

A.) A type of boolean values.

B.) A type of integers.

C.) A type of immutable strings.

D.) All of the above.

Continued on page 5

(18) [2 pts.] If the size of a value of the C type `double` is 64 bits, what is the value of `sizeof(double)`?

A.) 4.

B.) 8.

C.) 16.

D.) 64.

(19) [2 pts.] In C, space for a variable that is declared inside a function is allocated from

A.) Static memory.

B.) The call stack.

C.) The heap.

D.) Permanent storage.

(20) [2 pts.] In OCaml, which of the following data structures is always *mutable*?

A.) List.

B.) Record.

C.) Array.

D.) All of the above.

(21) [2 pts.] In C, which of the following data structures is always *immutable*.

A.) Record.

B.) Array.

C.) String.

D.) None of the above.

(22) [2 pts.] Which of the following data structures is *dynamic*?

A.) Record.

B.) Array.

C.) Linked list.

D.) All of the above.

Continued on page 6

(23) [2 pts.] Let us say that a type is *user designed* if the user defines the structure of the type during design time. In OCaml, which of the following data types is the most user designed?

A.) A reference type.

B.) A list type.

C.) A sum type.

D.) A product type.

(24) [2 pts.] Let us say that a type is *user designed* if the user defines the structure of the type during design time. In C, which of the following data types is the most user designed?

A.) A numeric type like `double`.

B.) A record type.

C.) An array type.

D.) A pointer type.

(25) [2 pts.] Unlike in OCaml, functions are not first-class values in C. Which of the following properties do functions have in OCaml but not in C?

A.) A function can be defined by recursion.

B.) A function can be defined within the definition of another function.

C.) A function can take another function as input.

D.) A function can return another function as output.

(26) [2 pts.] If `nestor` is the name of a pointer in C, which of the following is a selector for `nestor`?

A.) `*nestor`.

B.) `&nestor`.

C.) `nestor`.

D.) All of the above.

Continued on page 7

(27) [8 pts.] The *scalar multiple* of a real number r and a vector $V = (a, b)$ is the vector $r \cdot V = (r * a, r * b)$. Suppose the following OCaml phrase has been evaluated:

```
type vector = {x : float; y : float} ;;
```

Define an OCaml function named `scalar_multiply` of type

```
float -> vector -> vector
```

that implements scalar multiplication.

Answer:

```
let scalar_multiply r v =
  {x = r *. v.x; y = r *. v.y} ;;
```

Continued on page 8

(28) [8 pts.] Define an OCaml function named `concatenate_arrays` of type

```
int array -> int array -> int array
```

that takes as input two arrays x and y of type `int array` and returns as output an array z of type `int array` such that z is the concatenation of x and y. For example,

```
concatenate_arrays [|1;2;3|] [|5;6|] = [|1;2;3;5;6|]
```

should be true.

Answer:

```
let concatenate_arrays x y =
  let m = Array.length x in
  let n = Array.length y in
  let z = Array.create (m + n) 0 in
  for i = 0 to m - 1 do
    z.(i) <- x.(i)
  done ;
  for i = 0 to n - 1 do
    z.(m + i) <- y.(i)
  done ;
  z ;;
```

Continued on page 9

(29) [8 pts.] Suppose the following OCaml phrases have been evaluated:

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

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

A value of type `node` represents a binary tree. Write an OCaml function named `count` of type

```
node -> float -> int
```

such that `count n x` is the number of nodes in the binary tree represented by `n` whose data fields hold the value `x`. You may assume that `n` represents a *finite* binary tree. *Hint: Use recursion.*

Answer:

```
let rec count n x = match n with
  | Null -> 0
  | Normal r ->
      let a = (if r.data = x then 1 else 0) in
      a + (count r.left x) + (count r.right x) ;;
```

Continued on page 10

(30) [8 pts.] The *scalar multiple* of a real number r and a vector $V = (a, b)$ is the vector $r \cdot V = (r * a, r * b)$. Suppose the following C code has been evaluated:

```
typedef struct {
  double x;
  double y;
} vec_rec;

typedef vec_rec * vector;
```

Define a C function with prototype

```
vector scalar_multiply(double r, vector v);
```

that implements scalar multiplication. (The function should create a new vector, not modify the input vector.)

Answer:

```
vector scalar_multiply(double r, vector v) {
  vector v_new = malloc(sizeof(vec_rec));
  v_new->x = r * v->x;
  v_new->y = r * v->y;
  return v_new;
}
```

Continued on page 11

(31) [8 pts.] Define a C function with prototype

```
int * concatenate_arrays(int * x, int m, int * y, int n);
```

that takes as input two pointers x and y of type `int *` and two integers m and n of type `int` and returns as output a pointer z of type `int *` such that z points to an array of length $m + n$ that is the concatenation of the array of length m pointed to by x and the array of length n pointed to by y. For example, if a points to the array {1,2,3} and b points to the array {5,6}, then

`concatenate_arrays`$(a, 3, b, 2)$

points to the array {1,2,3,5,6}.

Answer:

```
int * concatenate_arrays(int * x, int m, int * y, int n) {
  int * z = malloc(sizeof(int[m + n]));
  int i = 0;
  for (i = 0; i < m; i++) {
    z[i] = x[i];
  }
  for (i = 0; i < n; i++) {
    z[m + i] = y[i];
  }
  return z;
}
```

Continued on page 12

(32) [8 pts.] Suppose the following C code has been evaluated:

```
typedef struct n_rec {
  double data;
  struct n_rec * left;
  struct n_rec * right;
} node_rec;

typedef node_rec * node;
```

Write a function in C with prototype

```
int count(node n, double x);
```

such that `count(n, x)` is the number of nodes in the binary tree represented by `n` whose data fields hold the value `x`. (When the value of `n` is `NULL`, let the value of `count(n, x)` be 0.) You may assume that `n` represents a *finite* binary tree. *Hint: Use recursion.*

Answer:

```
int count(node n, double x) {
  if (n == NULL) {
    return 0;
  }
  else {
    int a = (n->data == x ? 1 : 0);
    return a + count(n->left, x) + count(n->right, x);
  }
}
```

The End