**Name** ______________________________

**Student number** ______________________________

# CS 2SC3 and SE 2S03 Fall 2009

## Midterm Test Answer Key

Instructor: William M. Farmer

You have 50 minutes to complete this test consisting of 8 pages and 21 questions. You may use your notes and textbooks, but you may not use any calculators or other electronic devices. Answers to the first sixteen questions (1–16) are to be marked on the OMR scan sheet. Answer the last five questions (17–21) in the space provided on the exam. Do **NOT** use correction fluid on the exam or on the OMR scan sheet. Choose the *best* answer for the multiple choice questions. 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".

(1) [3 pts.] Without requirements there is no basis for judging the correctness or quality of a program. Is this statement true or false?

(a) [✓ True.]

(b) False.

(2) [3 pts.] OCaml does not technically have statements, but expressions of type `unit` serve as statements. Is this statement true or false?

(a) [✓ True.]

(b) False.

(3) [3 pts.] OCaml has a notation for defining functions that is equivalent to lambda notation. Is this statement true or false?

(a) [✓ True.]

(b) False.

(4) [3 pts.] Programming languages rarely have a precisely defined semantics. Is this statement true or false?

(a) [✓ True.]

(b) False.

(5) [3 pts.] In OCaml and most other programming languages, a nonterminating loop will eventually exhaust the memory allocated for the call stack. Is this statement true or false?

(a) True.

(b) [✓ False.]

(6) [3 pts.] The functional programming paradigm emphasizes the use of

(a) Mutable data structures.

(b) Iterative control structures.

(c) [✓ Side-effect free procedures.]

(d) All of the above.

(7) [3 pts.] The imperative programming paradigm emphasizes the use of

(a) Mutable data structures.

(b) Control structures.

(c) Assignment statements.

(d) [✓ All of the above.]

(8) [3 pts.] In OCaml, a reference of type `char ref` *cannot* be simulated by a

- (a) [✓ List.]
- (b) Record.
- (c) Array.
- (d) String.

(9) [3 pts.] Which of the following is an imperative statement in English?

- (a) The sun is hot today.
- (b) If I were omniscient, this midterm would be easy.
- (c) What great book did Euclid write?
- (d) [✓ Please answer all the questions.]

(10) [3 pts.] In OCaml, which of the following data structures holds a finite sequence of values?

- (a) List.
- (b) Array.
- (c) Stack.
- (d) [✓ All of the above.]

(11) [3 pts.] Which OCaml form is used to catch exceptions?

- (a) `match` $E$ `with` $P$.
- (b) [✓ `try` $E$ `with` $P$.]
- (c) `function` $N$ `->` $E$.
- (d) `let` $N$ `=` $E$.

(12) [3 pts.] In OCaml, a function is defined at

- (a) [✓ Design time.]
- (b) Compile time.
- (c) Run time.
- (d) None of the above.

(13) Which execution mode is useful for making programs portable?

- (a) Interpretation.
- (b) Compilation to native code.
- (c) [✓ Compilation to byte code.]
- (d) Compilation to a high-level language.

**Test continues on next page.**

(14) [3 pts.] A loop invariant can be used to

- (a) Prove properties about the loop.
- (b) Document the behavior of the loop.
- (c) Guide the design of a loop.
- (d) [✓ All of the above.]

(15) [3 pts.] In OCaml, the type

```
char array
```

is essentially the same as the type

- (a) `int array`.
- (b) [✓ `string`.]
- (c) `int -> char`.
- (d) `char list`.

(16) [3 pts.] Suppose `f` is defined in OCaml by

```
let f x = [] ;;
```

What is the type of `f`.

- (a) `'a list`.
- (b) `'int -> int list`.
- (c) `'a -> 'a list`.
- (d) [✓ `'a -> 'b list`.]

**Test continues on next page.**

(17) [6 pts.] Write a piece of OCaml code that defines a type named `trajan` of records having a immutable field named `awards` holding an array of strings and a mutable field named `alive` holding a boolean value.

**Answer**:

```
type trajan = {awards : string array; mutable alive : bool} ;;
```

(18) [6 pts.] Let `hadrian` be an OCaml function of type

```
int * int -> int.
```

Define an OCaml function named `curryed_hadrian` that is the Curryed form of `hadrian`.

**Answer**:

```
let curryed_hadrian x y = hadrian (x,y) ;;
```

(19) [10 pts.] Suppose the following types are declared in OCaml:

```
type sample_record =
    {x : float; y : float; z: float} ;;

type sample_tuple =
    (float ref) * (float ref) * (float ref) ;;
```

Define an OCaml function

```
marcus : sample_record -> sample_tuple
```

that, given a record *r* of type `sample_record` as input, returns a tuple *t* of type `sample_tuple` as output such that *r* and *t* represent the same ordered triple of rational numbers.

**Answer**:

```
let marcus r = ((ref r.x, ref r.y, ref r.z) : sample_tuple) ;;
```

(20) [15 pts.] In OCaml, define a *recursive* function

```
transform : int list -> float list
```

that transforms a list of values of type `int` to a list of values of type `float` such that the input list and output list represent exactly the same finite sequence of integers. You will receive no marks if you define this function using a loop instead of recursion.

**Answer**:

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

(21) [15 pts.] In OCaml, define a function

```
extract : float array -> float list
```

*using a loop* that extracts the positive values from an input array and returns them in an output list. You will receive no marks if you define this function using recursion instead of a loop. If `extract` works correctly and the order of the positive values in the input array is preserved in the output list, you will receive 4 bonus points.

**Answer**:

```
let extract input =
  let output = ref [] in
  for i = (Array.length input) - 1 downto 0 do
    if input.(i) > 0.
    then output :=  input.(i) :: !output
  done ;
  !output ;;
```

**Test ends here.**