CSL 102 Introduction to Computer Science

Major, Sem II 2006-07, Max 70, Time 2 hr

Name ______________________ **Entry No.** ______________________ **Group**

Note (i) This question paper has 4 pages

(ii) Write only in the space provided below each question including back of the sheets. Use the extra sheets for rough work.

(iii) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.

(iv) Note that the blank spaces do not carry equal marks. Marks will be awarded on the basis of overall correctness.

1. Write True/False against the following statements (1 × 5)

   (a) The entire program must reside in the main memory when it is executing.

   (b) Insertion sort runs faster in arrays than lists.

   (c) The length of a prefix expression is always of the form $2n - 1$ where n is a non-negative integer.

   (d) Any multiparamter function can be rewritten as a single parameter function in Ocaml.

   (e) The declaration `let rec x = let y = 0::x in 1::y ;;` implies a periodic structure `1::0::1::0 ..`

   Professor **OverConfident** claims that a circular list can be designed without using recursive structure or references. He says that arrays suffice - can you justify his claim ? Show how to represent the circular list $a_3 \Rightarrow a_2 \Rightarrow a_4 \Rightarrow a_1 \Rightarrow a_5 \Rightarrow a_3$ where the data fields are 5.4 , 2.0, 0.4 , 23.5 and 9.0 respectively; $a_1 \ldots a_5$ are the nodes of the circular list. Your scheme should provide for possible change in the links. You don't have to write in Ocaml syntax but draw a schematic showing the array(s). (5)

2. In the following function `root`, we compute the (positive) square root of a given floating point number x by an iterative method using the relation $x_{i+1} = \frac{1}{2} \cdot (x_i + \frac{x}{x_i})$ to compute the successive values of **x**. The program terminates when the x_i is within a predefined tolerance **eps**. The initial approximation is taken as $\frac{x}{2}$. Rewrite the function using recursion (without using `for` or `while` loops). Don't use any additional variables. (10)

```
let root x eps =  let y = ref ( x /. 2.0) in
                  if x < 0.0 then failwith "negative input"
                   else
                      while abs_float ( !y *. !y -. x ) > eps do
                           y := 0.5 *. ( !y +. x /. !y)
                       done ;
                    print_float !y ;;
```

$a^2 \geqslant k > (a-1)^2$

3. For a given integer $k \geq 1$, we want to compute an integer $a \geq 1$ such that (~~$a+1)^2 > k \geq$~~ a^2. For instance if $k = 11$, $a = 4$; for $k = 16$, $a = 4$. Complete the program below that finds a using binary search in the interval $[0, k/2]$. (10)

```
let introot k = if k =1 then 1 else if k =2 then 2 else
let rec binsrch(a,b)  =  (* sign changes between a and a+b *)

         if ____________________________________ then a else if b =1 then a+1

      else if ________________________________ then binsrch( a, b/2)

            else ________________________________________________

 in binsrch(0,k/2) ;;
```

4. To compute all the prime numbers in $[2, n]$, the *sieve of Eratosthenes* proceeds by eliminating all factors of 2, then all factors of 3 etc. Initially an array of size n is initialised to 1. In the i-th iteration, all the multiples of p_i (the i-th prime where $p_1 = 2$) are marked as 0 (not prime). In the end, i is prime if `a.(i) = 1`. (5+3+2)
(i) Complete the following program.

```
 let eratos n = let a = Array.create (n+1) 1 in (* index 0 to n *)

      for i = 2 to n do (* 2 is prime *)
       if a.(i) = 1 then
                   begin

            for j = _______ ____________to _____________________ do

                  ____________________ ________________ <- 0
                        done
                  end
              else () ;
            done ;;
```

(ii) What is the running time as a function of n ? (No proof needed)

(iii) Can you decrease the number of iterations of i ? (the outer for loop)

5. A point (x, y) is *dominated* by (x', y') if $x \leq x'$ and $y \leq y'$. Given a set S of n points (x_i, y_i), a point $p \in S$ is *maximal* if it is **not** dominated by any point in S. For example among $(3,4), (2,1), (4,0)$, $(3,4) and (4,0)$ are maximal. Complete the program below that computes all the maximal points of a given list `l` of points. The input list is given in **decreasing** order of x coordinates. The algorithm scans the points of `l` and the current point is maximal if and only if the y coordinate is larger than the y coordinates of all the points scanned before. The parameter `max` keeps track of the largest y

coordinate. The parameter **last** keeps track of the x-coordinate of the previous point. Note that the point with the largest x coordinate is always maximal - for simplicity you may assume that all points are distinct. (10)

```
let maximal l = let rec prefixmax lst last max = (*l is reverse sorted on 1st
                                                                  component *)
      match lst with
      [] -> [] |
      (a,b)::tail -> if a > last then failwith "not sorted"

         else if b > max then _______________________________________

               else prefixmax _______________________________________

        in match l with [] -> [] |

         (a,b)::tail -> ______________________________________________ ;;
```

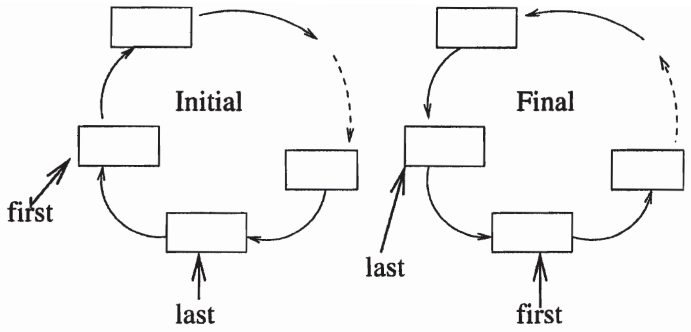
6. The value of $\sin x$ (in radians) is computed using the following series

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots (-1)^{i+1} \cdot \frac{x^{2i-1}}{(2i-1)!}$$

The series is truncated at $i = t$ where the value of the t-th term is a given tolerance `eps`. Write a function `seq_sum` that takes in two parameters - `eps` and a function `g : int -> float` that computes the i-th term of the series to compute the sum of a series upto a tolerance `eps`. Then write a function `sinfunc` to compute `sin x` using `seq_sum`. You may assume predefined functions `power (x,n)` that computes x^n and `fact n` that computes $n!$. (10)

```
let seq_sum
```

```
let sinfunc: ___________________________________ = function
```



7. Recall that a circular list was defined as follows
   type 'a inode = {info: 'a ; mutable next : 'a inode } ;;.
   You are given a circular list with **l** being the last element of the list pointing to the first element. We want to change the direction of the list and the original first **inode** becomes the last **inode** that points to the (original) last **inode** as the first node (see Figure).

   Complete the function below that maintains three consecutive inodes **prev** , **pres** , **next** and changes the **next** field appropriately to achieve the reversal of the direction. (10)

```
let reverse_list l = let first = l and pres = ref l.next and
                       prev = ref l and ahead = ref (l.next).next
                        in
              while not (first  == !pres) do

                (!pres).next             <- !prev ; (* sequence of update
                         is very critical otherwise you will lose the links*)

                ________________________   := !pres ;

                ___________________________:= !ahead ;

                 _________________________ := !ahead.next ;

               done;

             first.next <- _______________________ ;

            !______________ ;;(*returns the last element of the reversed list *)
```