

Name:

Entry No:

Group No:

### Major CSL 102

**NOTE:**

1. Attempt all seven questions.

2. Write your name and entry no on every sheet of the answer script.

**Time 2 Hrs**

**Max Marks 70**

Q No	Q 1	Q 2	Q 3	Q 4	Q 5	Q 6	Q 7	Total
MM	6	12	14	10	8	14	6	70

**Q1.** Write an **iterative** (tail recursive) function in SML for the following recursive function. The function is of the type  $f : N \rightarrow Z$ . It takes an input as a natural number and produces a positive integer. Show the run of your function on  $f(6)$ .

6

$$f(n) = \begin{cases} 1 & \text{if } n=0 \\ n-f(n-1) & \text{otherwise} \end{cases}$$

**Answer:**

```
fun f(n) =
  let
    fun fiter(n,p,c) =
      if c=n then p
      else fiter(n,c+1-p,c+1);
  in
    fiter(n,1,0)
  end;
```

```
f(6) = f_iter(6,1,0)
      = f_iter(6,0,1)
      = f_iter(6,2,2)
      = f_iter(6,1,3)
      = f_iter(6,3,4)
      = f_iter(6,2,5)
      = f_iter(6,4,6)
      => 4 (since c = n)
```

Name:

Entry No:

Group No:

**Q2.** As follows is an iterative function in SML to compute the  $x^n$ . What is the invariant in the function? Prove the correctness of the function using Principle of Mathematical Induction. Give the order of the time complexity of the function with explanation (formal proof is not required).

12

```
fun power(x, n) =
  let fun power_iter(p, q, m) =
        let fun odd(m) = (m mod 2 = 1);
            in
              if m=0 then p
              else if odd(m) then power_iter(p*q, q*q, m div 2)
              else power_iter(p, q*q, m div 2)
            end
        in power_iter(1, x, n)
      end;
```

**Answer:**

**Invariant:**

$(0 \leq m \leq n)$  and  $pq^m = p_0x^n$  (Here  $p_0=1$ )

**Correctness:**

If we show  $\text{power\_iter}(p, x, n) = px^n$  then  
we have  $\text{power}(x, n) = \text{power\_iter}(1, x, n) = x^n$

Basis:  $n = 0$   $\text{power\_iter}(p, x, n) = p = px^0$

Induction Hypothesis: for  $m$   $0 \leq m \leq n-1$   $\text{power\_iter}(p, x, m) = px^m$

Induction Step:

1. If  $n$  is odd ( $2k+1$ ) for  $k \geq 0$  (then  $n \text{ div } 2 = k$ )  
 $\text{power\_iter}(p, x, n) = \text{power\_iter}(p*x, x*x, n \text{ div } 2)$   
 $= p*x*(x*x)^{\text{ndiv}2}$  (by IH)  
 $= p*x*(x)^{\text{ndiv}2}*(x)^{\text{ndiv}2}$   
 $= p*x*(x)^{\text{ndiv}2}*(x)^{\text{ndiv}2}$   
 $= p*x*x^{n-1}$  ( $\text{ndiv}2 = k$  and  $2k+1=n$ )  
 $= p*x*x^{n-1}$   
 $= p*x^n$
2. If  $n$  is even ( $2k$ ) for  $k \geq 0$  (then  $n \text{ div } 2 = k$ )  
 $\text{power\_iter}(p, x, n) = \text{power\_iter}(p, x*x, n \text{ div } 2)$   
 $= p*(x*x)^{\text{ndiv}2}$  (by IH)  
 $= p*x^n$

**Complexity:**  $O(\log_2 n)$ : The algorithm does successive halving, that makes  $O(\log n)$  calls which have (2-3) operations of multiplication/division. Also if we consider  $n=2^r$  then a recurrence relation  $T(n) = T(n/2) + c$  can be written and the solution of which gives  $O(\log n)$

Name:

Entry No:

Group No:

**Q3.** Write a single function **SpecialSort** in SML to sort a list of integer elements such that the list is sorted in ascending order and there is no duplication of elements. You can however, define local functions (using let) within the **SpecialSort** function if need be. Please note that your program should not remove the duplicates before or after the sorting of the list as a pre or post processing step, it should be done during the sorting procedure. The algorithm should have a worst case time complexity of order  $O(n \log_2 n)$ .

14

**Answer:** *It is a modified merge sort, merge sort is the only sorting method that was studied that gives  $O(n \log n)$  worst case time complexity.*

```
fun SpecialSort([]) = []
  | SpecialSort ([x]) = [x]
  | SpecialSort (ls) =
    let
      fun split(ls) =
        let
          fun iter([],l1,l2,i) = (l1,l2)
            | iter(x::xs,l1,l2,i) =
              if (i mod 2 = 0) then
                iter(xs,x::l1,l2,i+1)
              else
                iter(xs,l1,x::l2,i+1)
          in
            iter(ls,[],[],0)
          end;
        val (l1,l2) = split(ls);
        fun merge([],l2) = l2
          | merge(l1,[]) = l1
          | merge(x::xs,y::ys) =
              if (x < y) then x::merge(xs,y::ys)
              else if (x > y) then y::merge(x::xs,ys)
              else x::merge(xs,ys)
        in
          merge(SpecialSort(l1),SpecialSort(l2))
        end;
```

Name:

Entry No:

Group No:

**Q4.** The  $k^{\text{th}}$  moment about the mean of the sequence  $[x_1, x_2, x_3, \dots, x_n]$  is given as follows.

$$k^{\text{th}} \text{ moment} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^k$$

Where

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$$

You are given the high order functions *mymap* and *myreduce* as defined below. Write an SML function **moment** (**L**, **k**) to compute  $k^{\text{th}}$  moment using these functions. In case you need any other function define it before you use it. The input to the function moment are the list of reals  $[x_1, x_2, x_3, \dots, x_n]$ .

10

```
exception Emptylist;

fun myreduce(F, nil) = raise Emptylist
  | myreduce(F, [a]) = a
  | myreduce(F, x::xs) = F(x, myreduce(F, xs));

fun mymap(F, nil) = nil
  | mymap(F, x::xs) = F(x) :: mymap(F, xs)
```

**Answer:**

```
fun power n x =
  if n=0 then 1.0
  else x*(power (n-1) x);

fun plus(x:real,y)=x+y;

fun length(nil) = 0.0
  | length(x::xs) = 1.0 + length(xs);

(* assuming myreduce and mymap available as above *)

fun moment(L, i) =
  let
    val n = length(L);
    val avg = myreduce(plus, L)/n
  in
    myreduce(plus, mymap(power i, mymap(fn(x)=>x-avg, L)))/n
  end;
```

Name:

Entry No:

Group No:

**Q5.** What does the following part of a JAVA program compute? A and B are arrays of integers. What is the time complexity? Can you do it better in terms of computational time? If so give the changed code and the improvement in time of computation.

8

```
int a;
for (i=0; i<n; i++)
{
    a = 0;
    for (int j=0; j<n; j++)
        a = a + A[j];
    B[i] = a/n;
}
```

**Answer:**

The above program computes the average of A[] and puts it in B[].

The complexity is  $O(n^2)$  for the reason there are two nested loops.

The program can be improved as follows:

```
int a;
a = 0;
for (i=0; i<n; i++)
    a = a + A[i];
a = a/n;
for (int i=0; i<n; i++)
    B[i] = a;
```

This makes the time complexity  $O(n)$ .

5

Name:

Entry No:

Group No:

**Q6.** An array **A** of size **N** contains three possible values: 1, 2, 3 in a random order. Give an **efficient algorithm** to partition the array such that all 1 entries come first, then all 2's and then all 3's, see the example given. You can use common constructs of Java like if-then-else, while, for loop, etc., to write your algorithm. Your algorithm should not use any additional array and should scan the array **A** only once and should have the time complexity of  $O(N)$ . The program should not count the number of 1's, 2's and 3's. The algorithm should be based on comparison and swapping, for swap or exchange you can define a function and use it. Give the invariant property of the loop used in your algorithm.

14

**Sample Input**

2	1	3	3	1	2	1	2	2
---	---	---	---	---	---	---	---	---

**Sample Output**

1	1	1	2	2	2	2	3	3
---	---	---	---	---	---	---	---	---

**Algorithm:**

We can consider three counters *lo*, *mid* and *hi*.  
Let the array *A* be defined 1 to *N*

```
lo = 1; mid =1; hi = N;
```

```
while (mid <= hi) do
  if a[mid] == 1 {
    swap a[lo] and a[mid]
    lo++; mid++;
  }
  else
  if a[mid] == 2
    mid++;
  else
    swap a[mid] and a[hi]
    hi--;
end
```

```
swap (a, b)
{ temp =a; a = b; b =temp; }
```

Invariant: At any stage algorithm should have

```
A[1...lo-1] = 1
A[lo...mid-1] = 2
A[hi+1...N] = 3
A[mid...hi] = ? (to be discovered)
```

That is

1	_____	lo	_____	mid	_____	hi	_____	N
	_____		_____		_____		_____	
←	---1---	→	←	-2-	→	←	---?---	→
←	-----3-----	→						

Name:

Entry No:

Group No:

**Q7.** A polynomial of degree  $n$  is defined as follows.

6

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \dots + a_nx^n$$

You have seen array implementation of polynomials in Java. When many of the coefficients are 0 (zero), i.e., the polynomial is sparse, it may not be efficient to use array to represent the polynomial. Therefore it is suggested to use linked lists to store/represent polynomials. That is, a polynomial in  $x$ , such as  $P(x) = 4.0x^{20} + 9.0x^{16} - 6.0x^5 + 3.0$  can be more efficiently stored as a linked list. Define the class Node in Java for the linked list to represent a polynomial. How can you find the first derivative of the polynomial using linked list? Give the steps for your algorithm in plain English (you are not required to write the Java code, if you want you can).

**Answer:**

```
class Node {
    int exp;
    double coeff;
    Node link;
    Node (c, e, Node n) {
        Coeff = c; exp = e; link = n;
    }
}
```

The derivative can also be represented by a list as the polynomial. The derivative list can be found:

1. Traverse the polynomial list and check if the  $e \geq 1$ ,
2. The node can be added (attached) to the derivative list with the data fields as  $\text{coeff} = c * e$ ; and  $\text{exp} = e - 1$ , where  $c$  and  $e$  correspond to the data (coeff and exp) of the current node in the polynomial list.
3. The may be reversed if need be (not required)