Computer Science Foundation Exam

May 5, 2006

Computer Science

Section 1B

Name: _____ KEY _____

SSN: _____

Q1		ANL	
Q2		KNW	
Q3		ANL,CMP	
Q4		CMP,ANL	
Q5		KNW,DSN	
Q6		CMP	
Total			

You have to do all the 6 problems in this section of the exam.
Partial credit cannot be given unless all work is shown and is readable.

Be complete, yet concise, and above all be neat.

1. **[4 pts]** It takes 72 ms for an algorithm to execute a task for 8 data elements. When the data size is increased 4 times, the algorithm takes 480 ms. What is the time complexity of the algorithm in terms of big-O?

This does not look like O(N).

Try if it is $O(N^2)$
8.8 /72 = 32.32/x
Yields x = 1152

Now try if it is O(N log N)
8 log 8 /72 = 32 log 32 /x
Yields x= 480.

Thus the algorithm is O(N logN)

[4 points for correct answer, 1 point for getting a wrong answer]

2. [**6 pts**]Consider the *circular array* based implementation of a queue with 30 slots. It uses two pointers, front and rear, both set to – 1 when the queue is empty.
Indicate the number of *EMPTY* slots on the queue for following combinations of front and rear indices. If any combination is meaningless indicate so.

i) front = 0, rear = 24 _____5

ii) front = 10, rear = 10 _____29

iii) front = – 1, rear = 29 _____meaningless

iv) front = 29 , rear = 29 _____ 29

v) front = 20, rear = 17 _____2

vi) front =5, rear = 20 _____14

[1 point for each correct answer]

3. [21 pts] Express the worst time complexities for following operations in terms of Big-O assuming most efficient algorithms are being used. Note that none of the operations are dependent on previous operations. *ALL operations are independent of each other.*

[1.5 points for each correct answer]

a) Given an array A with n elements in sorted order,

i)Searching for a specific element in A using an iterative algorithm O(logn)

ii) Searching for a specific element in A using a recursive algorithm O(logn)

iii) Adding 10 to the largest element of A. O(1)

b) Given a linked list L, with *m* nodes containing integers in sorted

i) Finding the contents of the middle element of L O(m)

ii) Inserting a new node in the 8^{th} position of L O(1)

iii) Adding 10 to the largest element of L O(m)

iv) Appending another linked list S containing *p* nodes to L O(m)

c) Given an arithmetic expression E with p operands, q arithmetic operators and n parentheses, checking if the parentheses in E are balanced O(p+q+n)

d) Given a queue Q with n elements, inserting 3 new elements in Q. O(1)

e) Given a circular linked list C, with n nodes and C pointing to the last node (last element of the list),

i) Deleting the first node of the list O(1)

ii) Inserting a new element at the end of the list O(1)

iii) Deleting the node pointed to by C O(n)

f) Given a stack S with p elements, Popping an element

i) when S is implemented using an array O(1)

ii) when S is implemented using a linked list O(1)

4. **[4 pts]** The number of steps involved in solving the Tower of Honoi problem for 30 disks is shown below. Making use of the recursive solution to this problem, work out the number of steps involved to solve the problem for 31 disks.

```
1,073,741,823

T(n) = 2 T(n-1)+1

T(31)=  2 .(1,073,741,823) + 1

     = 2,147,483,647
```

[4 points for getting the correct answer using the recurrence relation]

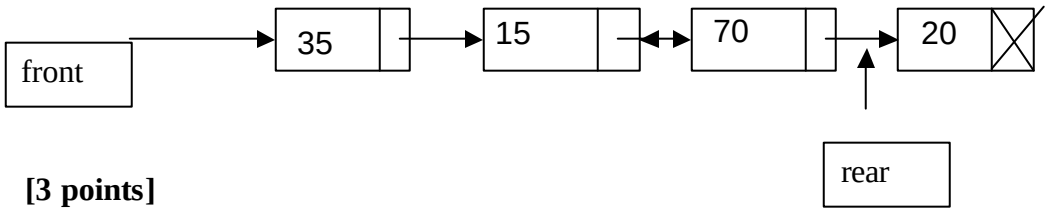
[1 point for getting the correct answer by any other means]

5. **[10 pts]** A queue is to be implemented using a linear linked list. Each node has the data structure:

```
struct node {
     int data;
     struct node * next;
};
```

Suppose following commands are given to such a queue :
enqueue 50
enqueue 35
enqueue 15
dequeue
enqueue 70
enqueue 20

a)Draw the linked list to show the status of the queue after ALL the commands have been executed. Show the pointers *front and rear.*



[3 points]

b) Write the dequeue function *void dequeue(struct node **front, struct node **rear, *d)* which returns the removed integer to the main program through the variable *d*. If the queue is empty, the function should return `-55.` Note double pointers are used, so the function does not return the list back to the main program.

```
void dequeue(s.n.**front, sn ** rear, *d)
{
  if (front==NULL)
        *d= -55;
```

[1 point for taking care of empty list]

```
  else
   {
        *d=(*front)->data;
        *front = (*front)->next;
```

[3 points for taking care of general case]

```
       if (*front == NULL)
            *rear = NULL;
```

[3 points for taking care of single node in the queue, by setting rear=NULL when front is NULL]

```
    }
 }
```

6. **[5 pts]** Given a circular linked list p, operate the following code

a) to indicate the order in which the elements of the list are being deleted

b) to indicate the final list returned by the function.

If the function is doing something meaningless, indicate so.

```
struct node *circular( struct node * p)
{
     struct node * r,temp;
     int j,count;
     temp= p;

     while(p->next ! = p)
     {
          r = p;
          for ( j=1; j < 4 ; j++)
              r  = r->next;
          temp =  r->next;
          r->next = temp->next;
          p = r->next;
          free ( temp);

     }
     return p;
}
```

Here is the list, with p pointing to 20.

20, 40, 5, 12, 50, 35, 70, 60, 80

SOLUTION: The code is supposed to delete every 5th element.

a) 50,20, 70, 12, 5, 35, 80, 40

[3 points for correct sequence, otherwise zero]

b) 60 **[2 points for indicating a single element here]**