You have to do all the 5 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. Tick the correct choices in the following. **[2pts for each correct answer]**

i) The worst case complexity of reversing a linked list containing n nodes is

a) $O(1)$  

b) $O(\log n)$  

c) $O(n)$  

d) $O(n \log n)$

ii) The worst case complexity of searching for a specific value in a linked list containing n nodes in sorted order is

a) $O(1)$  

b) $O(\log n)$  

c) $O(n)$  

d) $O(n \log n)$

iii) The time complexity of attaching a linked list containing j elements at the end of another linked list containing k elements would be

a) $O(j)$  

b) $O(k)$  

c) $O(j+k)$  

d) $O(jk)$

iv) A waiting line at a bank teller can be modeled using a

a) stack  

b) **queue**  

c) binary search tree

v) An infix expression is being converted to its postfix form using a stack. The character read from the expression is ‘-‘ and the stack contains the following elements.

```
/  +  /
```

If the character read from the expression is ‘-‘, the stack should look like

```
a b c d
```

```
/  -  -  /
```

```
-  +  +  -
```
As both operators have precedence equal to or greater than ‘-‘, both will be popped and replaced by ‘-‘. (Answer is (b))

2. Trace the following function when it is called from the main program through simple(7138)

```c
int simple ( int n) {
    if (n/10 == 0)
        return n;
    else
        return simple( n%10 + simple(n/10);
}
```

call stack:
simple(7138)
= simple (8 + simple(713))
= simple (8 + simple(3 + simple (71)))
= simple (8 + simple(3 + simple(1 + simple(7))))
= simple (8 + simple(3 + simple(1 + 7)))
= simple (8 + simple(3 + simple(8)))
= simple (8 + simple(11))
= simple (8 + simple(1 + simple(1)))
= simple (8 + 2)
= simple (0 + simple(1))
= 1

[Partial points may be awarded]
3. Write the recurrence relation for this function and work out the worst case time complexity for it.

```c
1 int power(a,n)
2 {
3     if(n == 1)  return a;
4     if ( n %2 == 0)
5         return power( a , n/2) * power (a, n/2);
6     else
7     return power( a , n/2) * power (a, n/2)* a;
8 }
```

The worst case is obtained by considering line 7

\[ T(n) = 2 \cdot T(n/2) + 4 \]

\[ T(1) = 1 \]

[ 4 pts for correct recurrence relations ]

Solution:

\[ T(n/2) = 2 \cdot T(n/4) + 4 \]

\[ T(n) = 4 \cdot T(n/4) + 2(4) + 4 \]

Or \[ T(n) = 4 \cdot T(n/4) + 3(4) \]

Now \[ T(n/4) = 2 \cdot T(n/8) + 4 \]

So \[ T(n) = 8 \cdot T(n/8) + 4(4) + 3(4) \]

Or \[ T(n) = 8 \cdot T(n/8) + 7(4) \]

This can also be written as

\[ T(n) = 2^3 \cdot T(n/2^3) + (2^3-1) \cdot (4) \]

[ 3 pts for correctly reducing the Right hand side ]

General case

\[ T(n) = 2^k \cdot T(n/2^k) + (2^k-1) \cdot (4) \]

[ 2 pts for correct general case ]

Let \[ n/2^k = 1 \]

\[ 2^k = n \]
\[ k = \log n \quad \text{[ 2 pts for correctly working out value of k ]} \]

Substituting the values in general case

\[ T(n) = n \ T(1) + (n - 1) \]

\[ T(n) = n + n - 1 \]

\[ T(n) = \mathcal{O}(n) \quad \text{[ 1 pt for correct time complexity ]} \]

4. Develop a RECURSIVE function which accepts an integer num, and prints its digits, one digit per line, in the order they appear in num. You can not convert the given integer to a string format. Thus given the integer 7354, it should print

7
3
5
4

\[ \text{[ 8 pts for correct solution,}
\]
\[ \text{Only 2 pts if solution prints digits in reverse order ]} \]

```c
void printdigits( int num)
{
    if (num == 0)
        return 0;
    else
    {
        printdigits( num/10);
        printf(“ %d \n”, num % 10);
    }
}
```
5. a) A circular list contains $m$ nodes with $KK$ pointing to the last node. Another circular list contains $n$ nodes with $MM$ pointing to its last node. Write a function which accepts $KK$ and $MM$, and attaches $MM$ to end of $KK$ to make a big circular list containing all the elements, and returns a pointer to the last element of this big list.

Thus given $KK$ and $MM$ as

```
KK
B -> C -> D -> F
```

```
MM
Q -> R -> S -> T
```

it should return the new circular list

```
B -> C -> D -> F -> Q -> R -> S -> T
```

5 b) What is the time complexity of this function in terms of $m$ and $n$?
struct node * combine ( struct node * KK, struct node * MM) 
{
    Struct node * temp1, *temp2;
    // save pointer to first element of KK
    temp1 = KK->next;
    temp2 = MM->next;
    // attach first element of MM to last element of KK
    KK->next = temp2;
    // let last element of MM point to first element of KK
    MM->next = temp1;
    // return pointer to last element of the big list
    Return KK;
}

The time complexity is O(1)