1. Answer the following questions.

(a) A single linked list contains n nodes. What is the time complexity of reversing this list in-place?

(b) Write a recursive C function to print all the elements present in a single linked list in reverse order.

(c) Construct a binary tree from the following sequence of tree traversal:

In-order traversal =*dbfeagcljhk*

Post-order traversal =*dfebgljkhca*

(d) While converting an infix expression: 6 + 5 * (8 * 5 – 9) to its postfix equivalent using stack, what will be the maximum number of symbols that appear in the stack at any point of time?

(e) In a max-heap of size n, what is the time complexity of extracting the maximum element and then inserting a new element?

(f) Suppose the depth of a fully complete binary tree is 6. If the level of the tree is starting from 0, how many nodes will be there in the tree?

(g) Write a modified bubble sort function whose time complexity is O (n) when it is applied to an already sorted array. (n is the number of elements in the array)

(h) What is the time complexity of the following code:

```
int num = 1, counter= N;
while (counter> 0) {
  num =num* i;
  counter= counter/2;
}
```

(i) Consider the following function applied to a single linked list with odd no. of nodes :

```
struct node * fun ( )
{
struct node *p, *q;
p=q=start;// start points to the first node of the list
while(q!=NULL && q->next!=NULL)
{
q=q->next->next;
p=p->next;
}
return p;
}
```

The code will return _____ node of the list.

(A) Last node

(B) Node before the last node

(C) Middle node

(D) None of options

(j) Suppose the key values {63, 22, 50, 23, 49, 57, 81} are stored in a hash table using the hash function: H(k)=K%7. Linear probing is used to resolve collision if present. After inserting the key values, the content of the hash table at index 4, 5, 6, and 7 are _____.

SECTION-B

2. (a) Write a C function of complexity O (log n) to count the number of elements present in a sorted array between values X and Y. The elements, to be counted, are greater than X and smaller than Y. X and Y are user inputs and may not be present in the array.

Example-1:

Input:

A[]= {7,11,13, 15, 18, 20, 21, 25, 37, 41, 49}

X=15, Y=49

Output: 6

Example-2:

Input:

A[]= {7,11,13, 15, 18, 20, 21, 25, 37, 41, 49}

X=17, Y=47

Output: 6

(b) Consider an array a[10]={10,74,19,80,27,93,33,5,12,24}

Perform insertion sort on the given array. Consider the case when a[8]=12, has to be inserted to its appropriate position in the sorted subset of the array

{a[0],a[1]....a[7]}. Mention the values of elements of the array at each step while reaching to the mentioned stage while performing insertion sort. Write down C function for insertion sort to sort an array in ascending order. Write the time complexity of insertion sort algorithm.

3. (a) You are given an array of integers, and your task is to find the maximum sub-array sum. A sub-array is defined as a contiguous subset of the array. Write a C function to find the maximum sub-array sum efficiently.

Example:

Input Array: [-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output: 6 (The maximum sub-array is [4, -1, 2, 1], and their sum is 6)

Note:

Your program should be able to handle both positive and negative numbers in the array. The maximum sub-array sum is the sum of elements in a sub-array such that the sum is as large as possible.

(b) What is the difference between a single linked list and double linked list representation? Write a C function to rotate a double linked list anti-clockwise by k nodes, where k is a given positive integer and smaller than the number of nodes in linked list.. For example, if the given linked list is A->B->C->D->E->F and k is 3, the list should be modified to D->E->F->A->B->C.

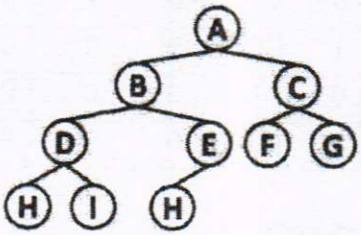
4. (a) Write a *non-recursive* C function to traverse a binary search tree in postorder. Define a stack ADT to be used in the function.

(b) Write a C function to construct a complete binary tree (using linked list) from a given level order traversal sequence of the tree.

Eg. :

Input: A B C D E F G H I H

Output:



5. (a) Define an input-restricted de-queue and an output-restricted de-queue with suitable diagrams and examples. Write a C code to check whether the content of the input-restricted de-queue is palindrome or not, by traversing each element only once.

(b) Write a C code to implement a priority queue using a binary heap. Explain it with an appropriate example.

6. (a) What is hashing? Explain collisions in hashing. Draw and explain the different hash tables with open addressing (quadratic probing, double hashing), and chaining by using hash functions h1(k) = k % 10, h2(k) = 7- (k % 7), as per requirements, with table size 10 (indexed from 0 to 9) by inserting the keys: 62, 53, 74, 32, 86, 23, 55 into the table in the given order. (Note: The hash function h2(k) is only used for double hashing technique)

(b) Consider that a visitor X visited the seven sister states of India: Assam (AS), Arunachal Pradesh (AP), Meghalaya (ML), Nagaland (NL), Manipur (MN), Mizoram (MZ)

and Tripura (TR) based on the following adjacency matrix **A**:

	AS	**AP**	**ML**	**NL**	**MN**	**MZ**	**TR**
AS	1	0	1	1	0	0	0
AP	0	0	0	1	1	0	0
ML	1	0	0	0	1	0	1
NL	1	1	0	0	0	1	0
MN	0	1	1	0	0	0	0
MZ	0	0	0	1	0	0	0
TR	0	0	1	0	0	0	0

Draw the undirected graph from matrix **A**, where states are vertices and edges indicate connected states (route of visitor X). Explain the Breadth-first Search (BFS) algorithm while applying it to traverse from start at **AS** to **MZ**. **AS** is the starting vertex and the goal is to reach MZ.

7. (a) What is an AVL tree? Write the difference between an AVL tree and a Binary Search Tree (BST) tree. Illustrate the steps in creating an AVL tree by inserting the sequence of elements: 11, 63, 21, 73, 85, 31, 42, 52, 75, 92, 67, 32, 19, 100, 17, 54 into the tree, starting with an empty tree. Show the AVL tree after deleting the node value 85 from the tree.

(b) Write a C function to construct a *modified* binary search tree where each node in the *modified* binary search tree has at most two children, a **left** child and a **right** child, with the **left** child containing values higher than the parent node and the **right** child containing values lesser than the parent node. Write a C code to traverse the tree to print the node key values in the descending order.