Name:_____________ Roll #: __________

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Data Structures**	**Course Code:**	**CS 2001**
Program:	**BS(CS, DS, SE, R)**	**Semester:**	**Fall 2023**
Duration:	**180 Minutes**	**Total Marks:**	**70**
Paper Date:	**23-dec- 2023**	**Page(s):**	**10**
Section:	**ALL**	**Section:**	
Exam:	**Final Exam**	**Roll No:**	

Instruction/Notes: Answer in the space provided. You can use rough sheets, which **will not be marked.** Do not use pencil or red ink to answer the questions**.** In case of confusion or ambiguity, make a reasonable assumption. Questions are not allowed.

Question 1: [CLO 1] **[Marks: 2*5]**

a. Explain how a single array can be converted into two stacks of equal size such that both can grow until there is some empty space in the array.

Initialize stack pointer for first stack from zero and for the second stack at size-1. Push operation will increment stack pointer for stack1 and decrement stack pointer for stack 2.

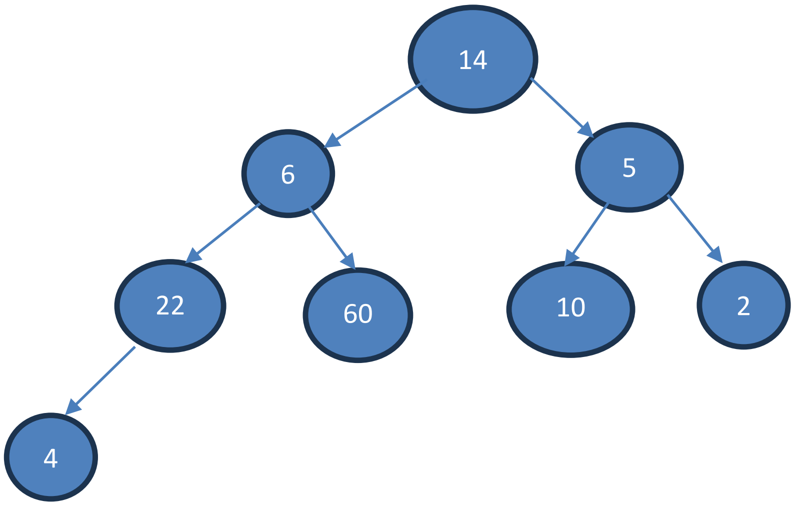
b. A full binary tree with 6 non-leaf nodes contains a maximum of how many nodes? Also provide the formula to determine maximum number of nodes at a specific level of Binary Tree.

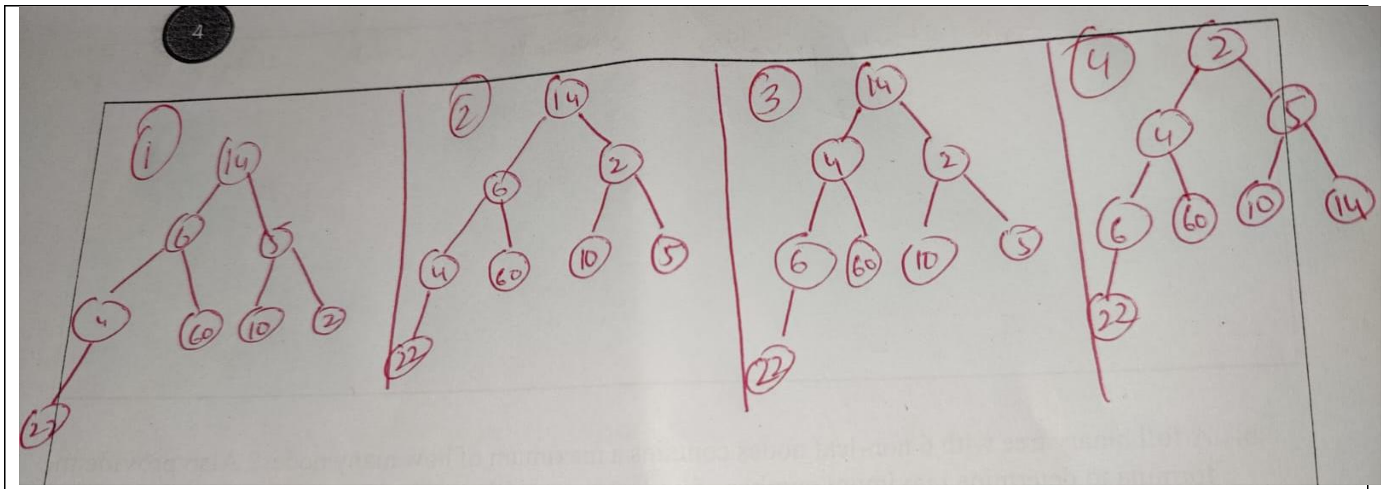
13 nodes in full binary tree with 6 non-leaf nodes (root is also a non leaf node)

2^L maximum number of nodes assuming Level number L starts at 0

Name:_____________ Roll #: __________

c. Build min heap from this given tree using BuildHeap method and show all of your working





d. Suppose an initially empty stack *S* has performed a total of 25 push operations, and 10 pop operations, 3 of which generated a StackEmpty exception that was caught and ignored. What is the current size of *S*?

25-7 =18 as for three operation nothing is poped from the stack

e. Which of the hash table collision-handling schemes could tolerate a load factor above 1 and which could not? Load factor is the ratio of number of elements present in the hash table and size of hash table.

Chaining

Question 2: [CLO 2] **[Marks:5+5+5+2.5+2.5]**

a. Give an estimate of T(N) for each line of the following code. Also, give time complexity (in Big-Oh notation). Compute the tight bounds.

```
long compute(int x) {
   long ans = 1;
   for (int i = 1; i <= x; i++)
      ans *= x;
   return ans;
}
void print(int n) {
   for (int i = 0; i < n; i++)
   {
      for (int j = 1; j < (n - i); j++)
         cout << "   ";

      for (int k = 0; k <= i; k++)
         cout << " " << compute(i) / (compute(k) * compute(i - k));

      cout << endl;
   }
   cout << endl;
}
```

O(n^3)

b. For each node u in an undirected graph, let twodegree[u] be the sum of the degrees of u's neighbors. Explain how to compute the entire array of twodegree[·] values in O(|V|+|E|) time, given a graph in adjacency list format.

First compute the degree of each node by traversing the adjacency list in the first pass. Again traverse the adjecancy list and now sum the degrees of all the neighbours of a node. This way twodegree of all the vertices can be computed in O(|V|+|E|) time.

c. Suppose you are given two circularly linked lists, *L* and *M* such that both the lists contain distinct integers. Describe an O(n) algorithm for telling if *L* and *M* are really the same list of nodes but with different starting points.

1. Traverse in list L until L ends and find the first element of List M in L.
2. Traverse both lists L and M until M ends(traverse L from the point where first element of M is found)
3. If all the elements match then return true otherwise return false

d. How long would it take to remove the ⌈log*n*⌉ smallest elements from a heap that contains *n* entries using the removeMin() operation?

$O(\log(n)*\log(n)) = O((\log(n))^2)$

e. Explain how to use an AVL tree to sort *n* elements in *O*(*n*log*n*) time in the worst case.

Insert all the n element in an AVL tree and then perform an inorder traversal

Question 3: [CLO 3] [Marks: 15+15]

a. Write a **recursive** C++ function bool IsHeap(…) in a **Binary tree** class, that determines if the Binary tree is a MIN-HEAP or not in **O(n) time.** Note its Binary tree (BT) not a BST.

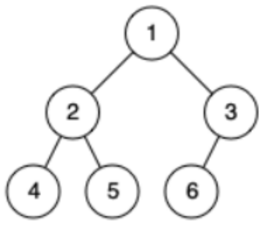
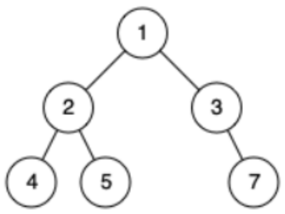
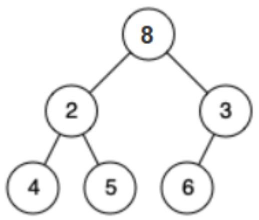
You code should sure that the Binary tree meets both requirements of heap to be classified as a binary min-heap.

i) Heap structure property (Complete Binary Tree)

ii) Heap Order property

Provide the code of any helper function that you use in your bool IsHeap(…) function. Write a wrapper function if needed.

class Bnode {
    public:
        T data;
        Bnode * left;
        Bnode * right;
};
	template <class T>
	class BinaryTree {
	    public:
	        BTree();
	        ~BTree();
	        bool IsHeap(…)
	    private:
	        Bnode * root;
	};

Example		
INPUT 1	INPUT 2	INPUT 3
 *bool IsHeap(...) returns true*	 *bool IsHeap(...) returns false* *//as structure property is violated*	 *bool IsHeap(...) returns false* *//as Heap order property is violated*

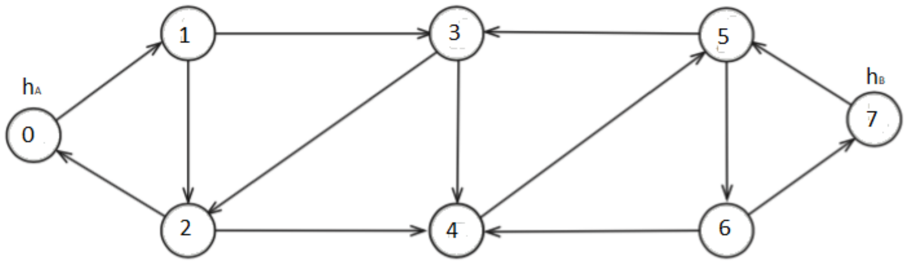
Name:_____________   Roll #: __________

```
int countNodes(struct Node* root)
       {
              if (root == NULL)
                     return (0);
              return (1 + countNodes(root->left) + countNodes(root->right));
       }
bool isComplete(Bnode* root, int index, int number_nodes)
{
              // An empty tree is complete
       if (root == NULL)
              return true;
       if (index >= number_nodes)
              return false;

       // Recur for left and right subtrees
       return (isComplete(root->left,2*index+1,number_nodes) && isComplete(root->right,
2*index + 2, number_nodes));
       }

bool isHeapProperty(Bnode* root)
{
       if (root->left == NULL && root->right == NULL)
              return true;

       if (root->right == NULL) {
              return (root->key >= root->left->key);
       }
       else {
              if (root->key >= root->left->key && root->key >= root->right->key)
                     return ((isHeapProperty(root->left)) && (isHeapProperty(root>right)));
              else
                     return false;
              }
}

bool IsHeap(Bnode* root) {
       int node_count = countNodes(root);
       int index = 0;
       if (isComplete(root, index, node_count) && isHeapProperty(root))
              return true;
       return false;
}

bool IsHeap(root) {
      return IsHeap(root) {


}
```

Name:_____________ Roll #: __________

b. Two spy, Ahmad and and Baber have been stationed in the enemy territory at houses h_A and h_B respectively. Ahmed needs to hand over a secret information to Baber. It is decided that Ahmed and Baber will meet each other at a third location, a hotel, where the information will be exchanged. The secret agency has already prepared a road map of the enemy territory. It contains the homes h_A, h_B, and the n hotels in the area: h1, h2, …, hn. This map formulates a graph where each location (houses and hotels) are its vertices and if one location is directly reachable via a road from the other then there is an edge between them. Each edge (road) poses a risk of being caught. The total risk of a path is simply the number of edges on that path.

The agency wishes to tell Ahmed and Baber which hotel, h, to meet at so that the total risk, for both of them combined, is minimized. Such a hotel would be the *safest hotel*. Note that both Ahmed and Baber would need to travel to h to make the delivery possible. The problem is that the map is too large for manual processing. Therefore, you have been tasked **to write a C++ program** that can find and return the id of the safest hotel given the graph G and locations h_A and h_B. ***Your goal is to write a function in Graph class that takes no more than O((|V|+|E|)) time to accomplish this task. If you use any helper function then also provide its code.*** You can assume that graph is represented as adjacency list and all the vertices are labelled 0 to |V|-1. Consider the example below: Vertex 0 is h_A and vertex 7 is h_B, here the safest hotel is vertex 3 and the minimum total risk is 4.



```
#include <list>
class Graph{
        int V;    // No. of vertices
        list<int> *adjList; //adjacency lists of neighbours (std list)
public:
        Graph(int V);   // Constructor
        … SafestHotel(…) // think of input parameter and return type
};
```

```
int safestHotel(int ha, int hb) {
             int* d1 = new int[V];
             int* d2 = new int[V];

             BFS(ha, d1);
             BFS(hb, d2);
             int min = 0;
             for (int i = 1;i < V;i++) {
                    if (d1[min] + d2[min] > d1[i] + d2[i])
                          min = i;
             }
             return min;
       }
```

Name:_____________ Roll #: __________

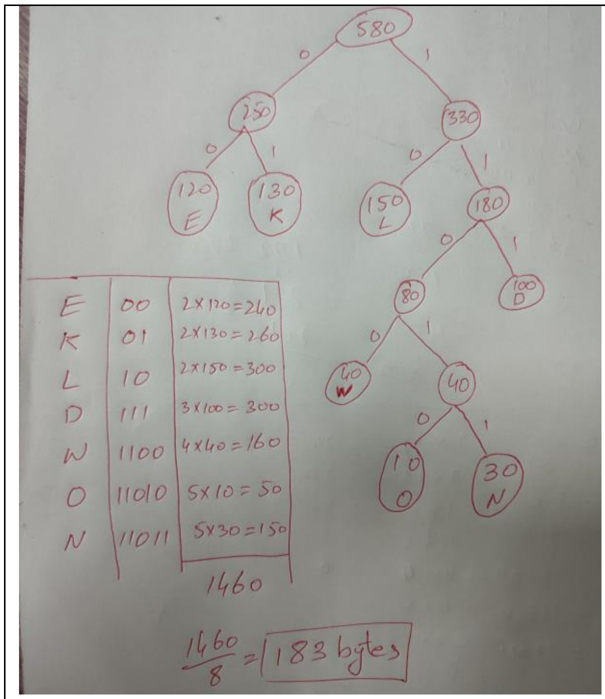
```
void BFS(int s, int* d) {
             bool* visited = new bool[V];
             for (int i = 0;i < V; i++)
                    visited[i] = false;
             d[s] = 0;
             queue<int> q;
             q.push(s);
             visited[s] = true;
             while (!q.empty()) {
                    int v = q.front();
                    q.pop();
                   list<int>::iterator it;
                    for (it = adjList[v].begin(); it != adjList[v].end();it++) {
                           int u = *it;
                           if (visited[u] == false) {
                                  visited[u] = true;
                                  d[u] = d[v] + 1;
                                  q.push(u);
                           }
                    }
             }
      }
```

Question 4: [CLO 4] **[Marks: 5+5]**

a. Suppose a file contains the following characters along with their frequencies.

D : 100, L : 150, E : 120, N : 30, K : 130, W : 40, O : 10

How many bytes will be required to store this file if Huffman encoding scheme is used. Show your complete working.



E	00	2x120=240
K	01	2x130=260
L	10	2x150=300
D	111	3x100=300
W	1100	4x40=160
O	11010	5x10=50
N	11011	5x30=150
		1460

$\frac{1460}{8} = \boxed{183 \text{ bytes}}$

b. Consider the following Hash tables, T1 and T2 of same sizes.
Hash function for table T1 is simple mod functions **t1 = hf1(key % T1_size)**
The hash function for table T2 is **t2= hf2(reverse_key % T2_size)** which reverses the key first and then takes mod with table size T2.
The keys **72,86,71,52,23 and 17** are inserted into an initially empty hash using open addressing and linear probing. After filling out both tables give a valid reason with hash function is better and why.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
T1	**72**	23	17		52		86	71

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
T2	**23**	71	52	72	86			17

Number of collisions faced by hf1 is 3 and hf2 is 1 so second hash function is better than first hash function

Name:______________ Roll #: ___________

Rough Sheet