National University of Computer and Emerging Sciences

Data Structures (CS2001)

Date: May 24, 2025

Course Instructor(s)

Dr. Aamir Wali, Mr. Rana Waqas,
Ms. Marwa Khan

Final Exam

Total Time: 3 hours
Total Marks: 55 marks
Total Questions: 3

________	________	____________________
Roll No	Section	Student Signature

- Attempt all questions in **sequence** on the answer sheet.
- Irrelevant or extra code will lead you to get **negative marks**.

Questions 1. Write codes of the following questions in C++. **[5+7+3 = 15 marks]**

Part a) You are given a directed weighted graph representing transactions between bank account holders.

- Each vertex represents a bank account holder (by name).
- Each directed edge represents a fund transfer from one person to another.
- The weight on the edge indicates the amount transferred.

For example, an edge from "A" to "B" with weight 250 means that "A" has transferred 250 rupees to "B". Your task is to identify the persons who have transferred the amount more than average transactions amount of all accounts.

Sample input:	**Sample output:**

names = {"A", "B", "C", "D"}
 	
graph = {
{ 0, 100, 200, 0},
{150, 0, 250, 100},
{100, 100, 0, 300},
{ 0, 100, 150, 0}
}
	C has transferred 500 rupees, which is above average
	B has transferred 500 rupees, which is above average

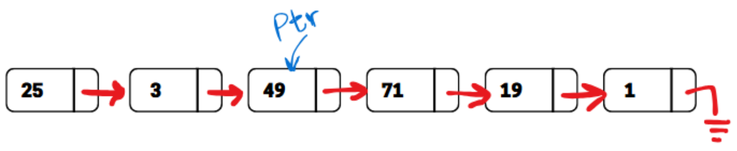
Part b) You are given a representation of a directory structure using a custom tree data structure, where each node represents a folder. Each folder may contain a name, size of folder, pointer to its sub-folder(s), pointer to its next sibling folder. This forms a tree-like structure similar to a real-world file system. Write a C++ program to traverse the directory structure and print the names of all folders along with their sizes, including subfolders, using recursion.

```
class Node {
public:
    string name;
    int size;
    Folder* subFolder;   // Points to the first sub-folder (child)
    Folder* next;        // Points to the next sibling folder
};
```

```
class Tree{
public:
    Node* root;
    void printFolderNames(Node*);
}
```

Sample directory representation	Sample output
Root, 512 ├── Documents, 400 │ ├── Projects, 310 │ └── Reports, 90 ├── Pictures, 100 │ └── Vacation, 100 └── Music, 12	Root, 512 Documents, 400 Projects, 310 Reports, 90 Pictures, 100 Vacation, 100 Music, 12

Part c) Suppose there is a singly linklist in which there is no other pointer than '**ptr**'. Your task is to write code snippet to delete the node to which this '**ptr**' is pointing in constant time complexity.

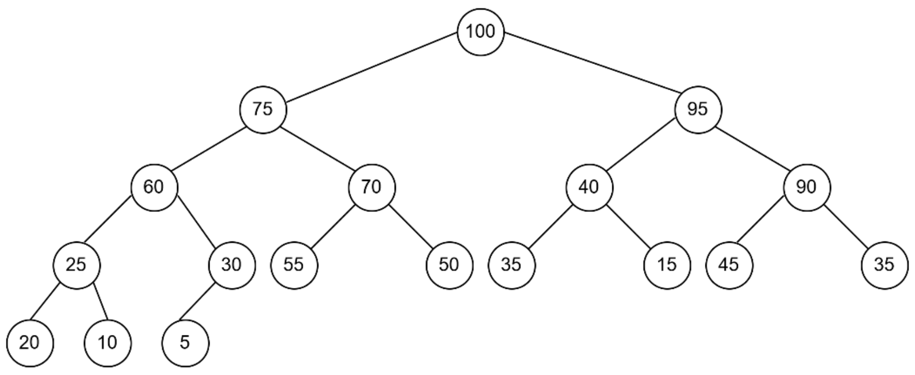


Question 2. **[5*4= 20 marks]**

a) Construct a binary tree by using postorder and inorder sequences given below.
Inorder: N, M, P, O, Q
Postorder: N, P, Q, O, M

b) Insert the values in the given order in a self-balancing tree that you have studied in your DSA course. Show all intermediate steps. **30, 20, 10, 40, 50, 25, 22, 24**

c) For the given max heap, apply insert operation for **(85)**. Show tree after every step.



d) After inserting 85 in part b perform deletion (). Show tree after every step.

Question 3. **[10+10 = 20 marks]**

Part a) Consider the following hash function that takes in a word and returns its hash code. We will use this function to determine the appropriate index in a hash table. In case of a collision, a **linear probing technique** is used.

```
int hash(char word[])
{
        int len = strlen(word);
        for (int i=0; i< len / 2; i++)
        {
                if( word[i] < 'm')
                        swap( word[i] ,  word [len-1-i]);
        }
        int val =0;
        for (int i=0; word[i];i++)
        {

                if(i+1 < len)
                        val = val + word[i] +  word [i+1] / 2 ;
                else
                        val =  val + word[i]-1;

        }
        return (val%7);
}
```

Store the following words in the hash table in the given order and apply linear probing when needed. Also, calculate the total number of collisions.
Words: lion, tiger, insect, bird, kangroo, donkey, cat

0	
1	
2	
3	
4	
5	
6	

Part b) Solve this question HERE on the question paper in provided space. Staple and submit the question paper along with your answer sheet.

Analyze and calculate the **Big-O time complexity** of the doSomething function. Clearly state the time complexities of all nested loops, statements and function calls (in column 2), and derive the overall complexity.

Code	**T(n)**
void doSomething(int n)	
{	
for (int i=1; i<n; i= i*2)	
for (int j=n; j>0; j = j/2)	
{	
if(i%2 == 0)	
hi(n);	
else	

bye(n)	
}	
}	
void hi(int n)	
{	
for (int i=0; i<5; i++)	
cout<”hi”;	
}	
void bye(int n)	
{	
for (int i=0; i<n; i++)	
cout<”bye”;	
}	