National University of Computer and Emerging Sciences

Data Structures (CS2001)

Sessional-II Exam

Date: April 10, 2025

Course Instructor(s)

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

Total Time: **1 hour**
Total Marks: **25**
Total Questions: **3**

_____ Roll No

_____ Section

_____ Student Signature

Do not write below this line

Attempt all questions on the answer sheet

CLO 2: Evaluate different data structures in terms of memory complexity and time.

Question 1. [5 marks]

For each of the following scenarios choose the "best" data structure from the following list or a combination of data structures: an unsorted array, linked list, DLL, circular LL, stack, queue. In each case, justify your answer briefly.

a. Suppose that a grocery store decided that customers who come first will be served first
b. A list must be maintained so that any element can be accessed randomly
c. A program needs to remember operations it performed in opposite order
d. The size of a file is unknown. The entries need to be entered as they come in. Entries must be deleted when they are no longer needed. It is important that structure has flexible memory management
e. A list must be maintained so that elements can be added to the beginning or end in O(1)

CLO 3: Design appropriate data structures to solve real world problems related to the program

Question 2. [10 marks]

A browser maintains visited URLs in a doubly linked list for back/forward navigation. A user wants to delete all history entries except the most recent visit to their bank's website (e.g., example-bank.com).

Task:

- Search for the last occurrence of the given URL in the list.
- Delete all other nodes before and after it while preserving the doubly linked structure.

```
class Node{
  public:
          string url;
          Node *next;
          Node *prev;
};
```

```
class DoublyLinkList{
    Node* head;
  public:
    void delAllHistory(string url){
          // provide the required code
    }
};
```

Question 3: [10 marks]

Write a recursive function **are_bst_similar(node *root1, node*root2)** that takes the roots of two binary search trees and returns True if the trees are exactly *similar* and False otherwise. Two BSTs are considered similar if they have:

- The same structure (i.e., nodes in the same positions).
- The same node values in the corresponding positions.

The node class has the usual members: int data, node* left, node* right

Sample Input:

Tree 1	Tree 2	Output
5 / \ 3 7 / 2	5 / \ 3 7 / 2	True
5 / \ 3 7	5 / \ 4 7	False