National University of Computer and Emerging Sciences

Lab Manual

for

Data Structures

Course Instructor	Mr. Razi Uddin
Lab Instructor(s)	Ms. Mateen Fatima Mr. Mamoona Akbar
Section	BSE-3C
Semester	FALL 2023s

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 04

Objectives:

After performing this lab, students shall be able to revise:

- ✓ single link list

Problem 1

Given a single link list of size n which is initialized by numbers from 1 to n in random order. User deletes four elements at random from list. Write a function that finds which elements are missing from the list.

E.g. singleLinkList

1->7->8->10->9->4->2->6->3->5

After user removed four elements from single link list.

singleLinkList

1->7->10->9->2->5

Missing elements

3->4->6->8

Problem 2

Link List is a dynamic data structure in which adjacency between elements is maintained using links/pointers. Implement a **singly linked list** class which stores integers. Your class declarations should look like:

```
class LinkedList;
class Node {
private:
int data;
Node*next;
};
class LinkedList { private:
Node* head;
public:
LinkedList(); // Default constructor
~LinkedList(); // Destructor
…
};
```

Implement the following LinkedList Functions:

a) bool insert (int val)

This function should insert a new value **(val)** into the linked list. The time complexity of this function should be **constant** i.e. **O(1)**. In other words, there should be no loop in this function. It returns true or false, if the data is inserted or not, respectively.

b) void display () : This function should display the contents of the linked list on screen.
c) int countNodes () : This function should determine (and return) the **count** of the nodes present in the linked list.
d) int returnFirst() : This function returns the data of the first node in the list.
e) int returnLast() : This function returns the data of the last node in the list.
f) **int pop()** : This function removes the first node and returns the data which was at that node.
g) bool insertNthNode(int data,int n): This function inserts the data at nth index in the list, and returns a bool value.
h) void combine (LinkedList& list1, LinkedList& list2) :

This function should combine the nodes of the two linked lists (**list1** and **list2**) into one list. All the nodes of the first list (**list1**) will precede (come before) all the nodes of the second list (**list2**). For example, if **list1** contain **{7 3 4 2}** and **list2** contains **{5 9}**, then after the function call **list3.combine(list1,list2)**, **list3** should contain **{7 3 4 2 5 9}** and **list1** and **list2** should be empty now.
You are NOT allowed to create any new node in this function. You can also assume that the LinkedList object on which this function is called is empty at the start of this function.

i) void shuffleMerge (LinkedList& list1, LinkedList& list2):

This function should shuffle-merge the nodes of the two linked lists (**list1** and **list2**) to make one list, by taking nodes alternately from the two lists. For example, if **list1** contains **{2 6 4}** and **list2** contains **{8 1 3}**, then after the function call **list3.shuffleMerge(list1,list2)**, **list3** should contain **{2 8 6 1 4 3}** and **list1** and **list2** should be empty now.
You are NOT allowed to create any new node in this function. You can also assume that the LinkedList object on which this function is called is empty at the start of this function.

j) void sortedInsert(head, node):
Write a **sortedInsert()** function which given a list that is sorted in increasing order, and a single node, inserts the node into the correct sorted position in the list.