Lab Final Exam

Marks

1. Write a program to sort an array of strings in lexicographic order using the merge sort algorithm. **10**

Input	Output
5 yellow apple children zzz chill	apple children chill yellow zzz
4 date cherry apple banana	apple banana cherry date

2. Implement a Doubly Linked-list of integers that maintains a **head** and a **tail.** Implement the following functions in your Doubly Linked-list. **10**
    - **insertHead(value)** : Inserts the value at the beginning of the linked-list. Expected Complexity O(1).
    - **insertTail(value)** : Inserts the value at the end of the linked-list. Expected Complexity O(1).
    - **insertMid(value)** : Inserts the value at the middle of the linked-list. Expected Complexity O(n).

3. In your implementation of question 2, add the following functions in your Doubly Linked-list class. **10**
    - **print()** : Prints the linked-list starting from head. Expected Complexity O(n).
    - **merge(LinkedList a)** : This function takes as input a LinkedList and merges the "LinkedList a" at the back of the current linked-list. Expected Complexity O(1).

Your implementation for problem 2 and 3 should look like this. You may write any extra functions that you need.

```
class Node{
        int value;
        Node* nxt;
        Node* prv;
};
```

```
class LinkedList{
        Node* head;
        Node* tail;
        LinkedList()
        {
                //Write your code
        }
        void insertHead(int value)
        {
                //Write your code
        }
        void insertTail(int value)
        {
                //Write your code
        }
        void insertMid(int value)
        {
                //Write your code
        }
        void print()
        {
                //Write your code
        }
        void Merge(LinkedList a)
        {
                //Merge a at the back of this linked-list
                //Write your code
        }
};
int main()
{
        LinkedList a;
        LinkedList b;

        a.insertHead(1);
        a.insertTail(5);
        a.insertMid(3);
        a.insertHead(0);
        a.insertTail(10);
        a.print(); // prints  0 1 3 5 10

        b.insertHead(10);
        b.insertTail(50);
        b.insertMid(30);
        b.insertHead(9);
        b.insertTail(100);
        b.print(); // prints  9 10 30 50 100

        a.Merge(b);
        a.print(); // prints  0 1 3 5 10 9 10 30 50 100
        b.print(); // prints  9 10 30 50 100
}
```

4. Write a program to check if a given bracket sequence is valid or not. The sequence will contain 3 types of brackets -> First Bracket ( ) , Second Bracket { } and Third Bracket [ ]. You can use builtin Stack for this problem. **10**

Input	Output
{[][]()(())}	Yes
{[][]()(()))}	No
{[](})	No

5. Implement a queue using a static array that supports enqueue(), dequeue(), and front() operations. Make the array size 100. **10**

6. You are given a ladder array of n integers. You need to sort it using a Deque. You can use builtin Deque for this problem. Expected Time Complexity is O(n).
A ladder array is an array that is increasing at first, then decreasing after that.
For example: [1,3,5,7,2,0] is a ladder array because 1 < 3 < 5 < 7 > 2 > 0. It is increasing till value 7, then it is decreasing after that. **10**

Input	Output
6 1 3 5 7 2 0	0 1 2 3 5 7
5 4 6 2 1 0	0 1 2 4 6

Hint: You just need to compare the values at the front and back of the Deque.

7. Implement a binary search tree that supports insertion and searching for a value.

Your implementation should look like this. You may write any extra functions that you need. **10**

```
class node{
public:
    int value;
    node* Left;
    node* Right;
};


class BST{
public:
    node *root;
    BST()
    {
        //Write your code here
    }
    void Insert(int value)
    {
           //Write your code here
    }
    bool Search(int value)
    {
        //Write your code here
    }
};
int main()
{
    BST bst;
    bst.Insert(10);
    bst.Insert(20);
    bst.Insert(25;
    bst.Insert(50);
    bst.Insert(8);
    bst.Insert(9);
    cout<<bst.Search(10)<<"\n"; //1
    cout<<bst.Search(9)<<"\n"; //1
    cout<<bst.Search(20)<<"\n"; //1
    cout<<bst.Search(60)<<"\n"; //0
    return 0;
}
```

8. Implement a MinHeap using a MaxHeap. Your implementation should look like this. **You are not allowed to write any other functions or variables.** **10**

```
class MinHeap{
public:
   MaxHeap mx;
   void insert(int x)
   {
        //Write your code here
   }
   void Delete(int idx)
   {
      //Write your code here
   }
   int getMin()
   {
      //Write your code here
   }
};
```

9. You are given a list of strings. You need to output for each string the previous index where it appeared. If it didn’t occur previously then output -1. Use STL Map for this problem. **10**

Input	Output
10 apple banana abcd apple abcd top abcd abcd apple banana	-1 -1 -1 0 2 -1 4 6 3 1

10. Given two sets, write a program to find the union of the two sets. You need to use STL Set for this problem. **10**

Input	Output

5
1 2 3 4 5
6
3 4 5 6 7 9
	1 2 3 4 5 6 7 9

The first array is [1,2,3,4,5] and the second array is [3,4,5,6,7,9]. Their union is [1, 2, 3, 4, 5, 6, 7, 9].