Answer Script

Question No. 01

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

Answer No. 01

Code:

```
#include<bits/stdc++.h>
using namespace std;

void Merge(vector<string>&ara, int l, int m, int r)
{
    int n1, n2;
    n1 = m - l + 1;
    n2 = r - m;

    vector<string> leftArray(n1);
    vector<string> rightArray(n2);

    for(int i=0; i<n1; i++)
        leftArray[i] = ara[l+i];
    for(int i=0; i<n2; i++)
        rightArray[i] = ara[m+1+i];

    int i=0, j=0, k=l;

    while(i<n1 && j<n2)
    {
        if(leftArray[i] < rightArray[j])
        {
            ara[k] = leftArray[i];
            i++;
        }
        else
        {
            ara[k] = rightArray[j];
            j++;
        }
        k++;
    }

    while(i<n1)
    {
        ara[k] = leftArray[i];
        i++;
        k++;
    }

    while(j<n2)
    {
        ara[k] = rightArray[j];
        j++;
        k++;
    }
}

void Merge_sort(vector<string>&ara, int l, int r)
{
    if(l>=r)
        return;

    int mid = l + (r - l)/2;

    Merge_sort(ara, l, mid);
    Merge_sort(ara, mid + 1, r);
    Merge(ara, l, mid, r);
}

void PrintLexicographicArray(vector<string>&ara, int n)
{
    for(int i=0; i<n; i++)
        cout<<ara[i]<<" ";
    cout << endl;
}

int main()
{
    int n;
    cin>>n;
    vector<string> ara(n);
    for(int i=0; i<n; i++)
        cin>>ara[i];

    Merge_sort(ara, 0, n-1);

    PrintLexicographicArray(ara, n);

    return 0;
}
```

Question No. 02

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).

Answer No. 02

Code:

```
#include<bits/stdc++.h>
using namespace std;

class Node
{
public:
    int value;
    Node* nxt;
    Node* prv;
};

class LinkedList
{
public:
    Node* head;
    Node* tail;
    int sz;

    LinkedList()
    {
        head = NULL;
        sz = 0;
    }

    Node* CreateNewNode(int value)
    {
        Node* newNode = new Node;
        newNode->value = value;
        newNode->nxt = NULL;
        newNode->prv = NULL;
    return newNode;
}

void insertHead(int value)
{
    sz++;
    Node* newNode = CreateNewNode(value);
    if(head == NULL)
    {
        head = newNode;
        tail = newNode;
        return;
    }
    Node* a = head;
    newNode->nxt = a;
    a->prv = newNode;
    head = newNode;
}

void insertTail(int value)
{
    sz++;
    Node* newNode = CreateNewNode(value);
    if(head == NULL)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        newNode->prv = tail;
        tail->nxt = newNode;
        tail = newNode;
    }
}

void insertMid(int value)
{
    if(sz == 0)
    {
        insertHead(value);
        return;
    }
    int mid = (sz / 2)-1;
    Node* a = head;
```

```
        int cur_index = 0;
        while(cur_index != mid)
        {
            a = a->nxt;
            cur_index++;
        }
        Node* newNode = CreateNewNode(value);
        newNode->nxt = a->nxt;
        newNode->prv = a;

        Node* b = a->nxt;
        b->prv = newNode;
        a->nxt = newNode;
        sz++;
    }

//    void print()
//    {
//        Node* a = head;
//        while(a != NULL)
//        {
//            cout<<a->value<<" ";
//            a = a->nxt;
//        }
//        cout<<"\n";
//    }
};

int main()
{
    LinkedList a;

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

Question No. 03

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
}
```

Answer No. 03

Code:

```
#include<bits/stdc++.h>
using namespace std;

class Node
{
public:
   int value;
   Node* nxt;
   Node* prv;
};

class LinkedList
{
public:
   Node* head;
   Node* tail;
   int sz;
```

```
LinkedList()
{
    head = NULL;
    sz = 0;
}

Node* CreateNewNode(int value)
{
    Node* newNode = new Node;
    newNode->value = value;
    newNode->nxt = NULL;
    newNode->prv = NULL;
    return newNode;
}

void insertHead(int value)
{
    sz++;
    Node* newNode = CreateNewNode(value);
    if(head == NULL)
    {
        head = newNode;
        tail = newNode;
        return;
    }
    Node* a = head;
    newNode->nxt = a;
    a->prv = newNode;
    head = newNode;
}

void insertTail(int value)
{
    sz++;
    Node* newNode = CreateNewNode(value);
    if(head == NULL)
    {
        head = newNode;
        tail = newNode;
    }
    else
    {
        newNode->prv = tail;
        tail->nxt = newNode;
        tail = newNode;
    }
}

void insertMid(int value)
{
    if(sz == 0)
    {
        insertHead(value);
        return;
    }
    int mid = (sz / 2)-1;
    Node* a = head;
    int cur_index = 0;
    while(cur_index != mid)
    {
        a = a->nxt;
        cur_index++;
    }
    Node* newNode = CreateNewNode(value);
    newNode->nxt = a->nxt;
    newNode->prv = a;

    Node* b = a->nxt;
    b->prv = newNode;
    a->nxt = newNode;
    sz++;
}

void print()
{
    Node* a = head;
    while(a != NULL)
    {
        cout<<a->value<<" ";
        a = a->nxt;
    }
    cout<<"\n";
}

void Merge(LinkedList a)
{
    if(a.head == NULL)
        return;
    if(head == NULL)
    {
      head = a.head;
      tail = a.tail;
    }
    else
    {
      tail->nxt = a.head;
      a.head->prv = tail;
      tail = a.tail;
    }

    sz += a.sz;
  }
};

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

  return 0;
}
```

Question No. 04

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

Answer No. 04

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   string s;
   cin>>s;

   stack<char> st;

   for(int i=0; i<s.size(); i++)
   {
      char ch = s[i];

      if(ch == '(' || ch == '{' || ch == '[')
         st.push(ch);
      else if(ch == ')' && !st.empty() && st.top() == '(')
         st.pop();
      else if(ch == '}' && !st.empty() && st.top() == '{')
         st.pop();
      else if(ch == ']' && !st.empty() && st.top() == '[')
         st.pop();
      else
         st.push(ch);
   }

   if(st.empty())
```

```
        cout<<"Yes\n";
    else
        cout<<"No\n";

    return 0;
}
```

Question No. 05

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

Answer No. 05

Code:

```
#include<bits/stdc++.h>
using namespace std;

const int MAX_SIZE = 100;

class Queue{
public:
    int a[MAX_SIZE];
    int l, r;
    int sz;

    Queue()
    {
        l = 0;
        r = -1;
        sz = 0;
    }

    void enqueue(int value)
    {
        if(sz == MAX_SIZE)
        {
            cout<<"Queue is full!\n";
            return;
        }

        r++;
        if(r == MAX_SIZE)
        {
            r = 0;
        }
        a[r] = value;
        sz++;
    }

    void dequeue()
```

```
   {
      if(sz == 0)
      {
         cout<<"Queue is empty!\n";
         return;
      }
      l++;
      if(l == MAX_SIZE)
      {
         l = 0;
      }
      sz--;
   }

   int front()
   {
      if(sz==0)
      {
         cout<<"Queue is empty!\n";
         return -1;
      }
      return a[l];
   }
};

int main()
{
  Queue q;

  q.enqueue(5);
  q.enqueue(6);
  q.enqueue(7);
  cout<<q.front()<<"\n";
  q.dequeue();
  q.dequeue();
  cout<<q.front()<<"\n";
}
```

Question No. 06

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.

Answer No. 06

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n;
    cin>>n;

    vector<int> ara(n);
    for(int i=0; i<n; i++)
        cin>>ara[i];

    deque<int> dq;

    for(int i=0; i<n; i++)
        dq.push_back(ara[i]);


    while(!dq.empty())
    {
        if(dq.front() > dq.back())
        {
            cout<<dq.back()<<" ";
            dq.pop_back();
        }
        else
        {
            cout<<dq.front()<<" ";
            dq.pop_front();
        }
    }

    return 0;
}
```

Question No. 07

7. Implement a binary search tree that supports insertion and searching for a value. **10**

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

```
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;
}
```

Answer No. 07

Code:

```
#include<bits/stdc++.h>
using namespace std;

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

class BST
{
public:
    node *root;

    BST()
    {
        root = NULL;
    }

    node* CreateNewNode(int value)
    {
        node* newNode = new node;
        newNode->value = value;
        newNode->Left = NULL;
        newNode->Right = NULL;
        return newNode;
    }

    void Insert(int value)
    {
        node* newNode = CreateNewNode(value);
        if(root == NULL)
        {
            root = newNode;
            return;
        }

        node* cur = root;
        node* prev = NULL;
        while(cur != NULL)
        {
            if(newNode->value > cur->value)
            {
                prev = cur;
                cur = cur->Right;
            }
            else
            {
                prev = cur;
                cur = cur->Left;
            }
        }

        if(newNode->value > prev->value)
        {
            prev->Right = newNode;
        }
        else
        {
            prev->Left = newNode;
        }
    }

    bool Search(int value)
    {
        node* cur = root;

        while(cur != NULL)
        {
            if(value > cur->value)
                cur = cur->Right;
            else if(value < cur->value)
                cur = cur->Left;
            else
                return true;
        }
        return false;
    }

};

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;
}
```

Question No. 08

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
    }
};
```

Answer No. 08

Code:

```
#include<bits/stdc++.h>
using namespace std;

class MaxHeap
{
public:
    vector<int> nodes;
    MaxHeap()
    {

    }

    void up_heapify(int idx)
    {
        while(idx > 0 && nodes[idx] > nodes[(idx-1)/2])
        {
```

```
        swap(nodes[idx], nodes[(idx-1)/2]);
        idx = (idx-1)/2;
    }
}

void Insert(int x)
{
    nodes.push_back(x);
    up_heapify(nodes.size()-1);
}

void down_heapify(int idx)
{
    while(1)
    {
        int largest = idx;
        int l = 2*idx + 1;
        int r = 2*idx + 2;
        if(l<nodes.size() && nodes[largest] < nodes[l])
            largest = l;
        if(r<nodes.size() && nodes[largest] < nodes[r])
            largest = r;

        if(largest == idx)
            break;

        swap(nodes[idx], nodes[largest]);
        idx = largest;
    }
}

void Delete(int idx)
{
    if(idx >= nodes.size())
        return;
    swap(nodes[idx], nodes[nodes.size()-1]);
    nodes.pop_back();
    down_heapify(idx);
}

int getMax()
{
    if(nodes.empty())
    {
        cout<<"Heap is empty! ";
```

```
            return -1;
        }
        return nodes[0];
    }


    void PrintHeap()
    {
        for(int i=0; i<nodes.size(); i++)
        {
            cout<<nodes[i]<<" ";
        }
        cout<<"\n";
    }
};

class MinHeap
{
public:
    MaxHeap mx;

    void insert(int x)
    {
        mx.Insert(-x);
    }

    void Delete(int idx)
    {
        mx.Delete(idx);
    }

    int getMin()
    {
        int minValue;
        minValue = mx.getMax();

        if(!mx.nodes.empty())
            return -minValue;
        else
            return -1;
    }

//    void print()
//    {
//        mx.PrintHeap();
//    }
};


int main()
{
    MinHeap heap;

    heap.insert(4);
    heap.insert(30);
    heap.insert(9);
    heap.insert(7);
    heap.insert(1);
//    heap.print();  ///1 4 9 30 7
    cout<<heap.getMin()<<"\n"; ///1
    heap.Delete(0);
//    heap.print();  ///4 7 9 30
    cout<<heap.getMin()<<"\n"; ///4

    return 0;
}
```

Question No. 09

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

Answer No. 09

Code:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int n;
   cin >> n;

   map<string, int> index;

   for(int i=0; i<n; i++)
   {
      string s;
      cin >> s;
      if(index.count(s))
      {
         cout<<index[s]<<endl;
      }
      else
      {
         cout<<-1<<endl;
      }
```

```
        index[s] = i;
    }

    return 0;
}
```

Question No. 10

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].

Answer No. 10

Code:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int n, m;
   set<int> st;

   cin>>n;
   for(int i=0; i<n; i++)
   {
      int x;
      cin>>x;
      st.insert(x);
   }

   cin>>m;
   for(int i=0; i<m; i++)
   {
      int x;
      cin>>x;
      st.insert(x);
   }

   for(auto it: st)
      cout<<it<<" ";
return 0;
}
```