Answer Script

Question No. 01

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

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

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

vector<string>merge_sort(vector<string>temp)
{
   if(temp.size() <= 1)return temp;

   int mid = temp.size()/2;
   vector<string>left;
   vector<string>right;
   for(int i = 0; i<mid; i++)
      left.push_back(temp[i]);
   for(int j = mid; j<temp.size(); j++)
      right.push_back(temp[j]);

   vector<string>sorted_left = merge_sort(left);
   vector<string>sorted_right = merge_sort(right);
   vector<string>sorted_temp;
```

```
    int ind1 = 0;
    int ind2 = 0;

    for(int k=0; k<temp.size(); k++)
    {
        if(ind1 == sorted_left.size()){
            sorted_temp.push_back(sorted_right[ind2]);
            ind2++;
        }

        else if(ind2 == sorted_right.size()){
            sorted_temp.push_back(sorted_left[ind1]);
            ind1++;
        }

        else if(sorted_left[ind1] <= sorted_right[ind2]){
            sorted_temp.push_back(sorted_left[ind1]);
            ind1++;
        }
        else if(sorted_right[ind2] < sorted_left[ind1]){
            sorted_temp.push_back(sorted_right[ind2]);
            ind2++;
        }

    }

    return sorted_temp;
}

int main()
{
    int t;cin>>t;
    vector<string>input(t);
```

```
    for(int i=0; i<t; i++){
        cin>>input[i];
    }
    vector<string> ans= merge_sort(input);
    for(int i=0; i<ans.size(); i++)
        cout<<ans[i]<<" ";

    cout<<endl;
    return 0;
}
```

Question No. 02

Implement a Doubly Linked-list of integers that maintains a **head** and a **tail.** Implement the following functions in your Doubly Linked-list.

- **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

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

class node{
public:
    node *prev;
    int data;
    node *next;
};

class DLL{
```

```
public:
   node *head;
   node *tail;
   int sz;

   DLL()
   {
      head = NULL;
      tail = NULL;
      sz = 0;
   }

   node *CreateNode(int value)
   {
      sz++;
      node *newnode = new node;
      newnode->prev = NULL;
      newnode->data = value;
      newnode->next = NULL;
      return newnode;
   }

   void insertHead(int value)
   {
      node *temp = CreateNode(value);
      if(head == NULL)
      {
         head = temp;
         tail = temp;
         return;
      }
      temp->next = head;
      head->prev = temp;
      head = temp;
```

```
}

void insertTail(int value)
{
   node *temp = CreateNode(value);
   if(head == NULL)
   {
      head = temp;
      tail = temp;
      return;
   }
   temp->prev = tail;
   tail->next = temp;
   tail = temp;
}

void insertMid(int value)
{
   if(sz < 2){
      insertTail(value);
      return;
   }
   int ind = 0;
   node *temp = head;
   while(ind < (sz/2)-1){
      ind++;
      temp = temp->next;
   }

   node *newnode = CreateNode(value);
   newnode->next = temp->next;
   temp->next->prev = newnode;
   temp->next = newnode;
   newnode->prev = temp;
```

```
    }

    void print()
    {
        node *temp = head;
        while(temp != NULL){
            cout<<temp->data<<" ";
            temp = temp->next;
        }
        cout<<endl;
    }
};

int main()
{
    DLL dl;
    dl.insertHead(5);
    dl.insertHead(5);
    dl.insertHead(5);
    dl.insertTail(3);
    dl.insertTail(8);
    dl.insertTail(9);
    dl.insertMid(1000);
    dl.print();

    return 0;
}
```

Question No. 03
In your implementation of question 2, add the following functions in your Doubly Linked-list class.

- **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).

Answer No. 03

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

class node{
public:
   node *prev;
   int data;
   node *next;
};

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

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

   node *CreateNode(int value)
   {
      sz++;
      node *newnode = new node;
```

```
    newnode->prev = NULL;
    newnode->data = value;
    newnode->next = NULL;
    return newnode;
}

void insertHead(int value)
{
    node *temp = CreateNode(value);
    if(head == NULL)
    {
        head = temp;
        tail = temp;
        return;
    }
    temp->next = head;
    head->prev = temp;
    head = temp;
}

void insertTail(int value)
{
    node *temp = CreateNode(value);
    if(head == NULL)
    {
        head = temp;
        tail = temp;
        return;
    }
    temp->prev = tail;
    tail->next = temp;
    tail = temp;
}
```

```
void insertMid(int value)
{
   if(sz < 2){
      insertTail(value);
      return;
   }
   int ind = 0;
   node *temp = head;
   while(ind < (sz/2)-1){
      ind++;
      temp = temp->next;
   }

   node *newnode = CreateNode(value);
   newnode->next = temp->next;
   temp->next->prev = newnode;
   temp->next = newnode;
   newnode->prev = temp;

}

void print()
{
   node *temp = head;
   while(temp != NULL){
      cout<<temp->data<<" ";
      temp = temp->next;
   }
   cout<<endl;
}

void Merge(LinkedList temp)
{
   tail->next = temp.head;
```

```
        temp.head->prev = tail;
    }
};

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

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.

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

Answer No. 04

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

int main()
{
   stack<int>st;
   string input;cin>>input;

   for(int i=0; i<input.size(); i++)
   {
      char now = input[i];
      if(now == '(' || now == '{' || now == '['){
         st.push(now);
      }
      else
      {
         if(st.empty()){
            st.push(now);
            break;
         }
```

```
        else
        {
            if(now == ')' && st.top() == '('){
                st.pop();
            }
            else if(now == '}' && st.top() == '{'){
                st.pop();
            }
            else if(now == ']' && st.top() == '['){
                st.pop();
            }
            else
                break;
        }
    }
}

if(st.empty())cout<<"YES"; else cout<<"NO";

return 0;
}
```

Question No. 05
Implement a queue using a static array that supports enqueue(), dequeue(), and front() operations. Make the array size 100.
Answer No. 05

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

const int MAX_SIZE = 100;
```

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

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

    void enqueue(int value)
    {
        if(r+1 >= MAX_SIZE)
        {
            cout<<"NO SPACE !!"<<endl;
            return;
        }
        else
        {
            r++;
            arr[r] = value;
            sz++;
        }
    }

    void dequeue()
    {
        if(r<l){
            cout<<"Already Empty !!"<<endl;
            return;
        }
        else
```

```
        {
            arr[l] = 0;
            l++;
            sz--;
        }
    }

    void Front()
    {
        if(r<l){
            cout<<"EMPTY!!"<<endl;
            return;
        }
        cout<<"FRONT : "<<arr[l]<<endl;
    }

    void print()
    {
        if(sz==0){
            cout<<"Nothing in Queue"<<endl;
            return;
        }
        for(int i=l; i<=r; i++)
            cout<<arr[i]<<" ";
        cout<<endl;
    }
    void Size(){
        cout<<"SIZE : "<<sz<<endl;
    }

};

int main()
{
```

```
    Queue qu;
    qu.enqueue(10);
    qu.enqueue(20);
    qu.enqueue(30);
    qu.enqueue(40);
    qu.enqueue(50);
    qu.Front();
    qu.print();
    qu.Size();

    qu.dequeue();
    qu.Front();
    qu.print();
    qu.Size();

    qu.dequeue();
    qu.Front();
    qu.print();
    qu.Size();

    qu.enqueue(1000);
    qu.Front();
    qu.print();
    qu.Size();

    return 0;
}
```

Question No. 06

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.

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

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

int main()
{
   int t;cin>>t;
   deque<int>input;
   deque<int>ans;
   for(int i=0; i<t; i++)
   {
      int temp;cin>>temp;
      input.push_back(temp);
   }
   while(!input.empty()){
      if(input.front() < input.back()){
         ans.push_back(input.front());
         input.pop_front();
      }
      else
      {
         ans.push_back(input.back());
         input.pop_back();
      }
   }

   for(int j=0; j<t; j++){
```

```
        cout<<ans.front()<<" ";
        ans.pop_front();
    }

    return 0;
}
```

Question No. 07

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

Answer No. 07

```
#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 *CreatNode(int value)
    {
        node *newnode = new node;
        newnode->value = value;
        newnode->Left = NULL;
        newnode->Right = NULL;
        return newnode;
    }

    void Insert(int value)
```

```
{
    node *newnode = CreatNode(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 *temp = root;
    while(temp != NULL)
    {
        if(value > temp->value)
        {
            temp = temp->Right;
        }
        else if(value < temp->value)
        {
```

```
            temp = temp->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

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

```
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

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

class MaxHeap{
public:
   vector<int>nodes;

   MaxHeap()
   {

   }

   void up_heapify(int ind)
   {
      while(ind > 0 &&  nodes[ind] > nodes[(ind-1)/2])
      {
         swap(nodes[ind], nodes[(ind-1)/2]);
         ind = (ind-1) / 2;
      }
   }

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

   void down_heapify(int idx)
   {
      while(true)
      {
         int largest = idx;
         int left = 2*idx + 1;
```

```
        int right = 2*idx + 2;
        if(left < nodes.size() && nodes[largest] < nodes[left]){
            largest = left;
        }
        if(right < nodes.size() && nodes[largest] < nodes[right]){
            largest = right;
        }
        if(largest == idx){
            break;
        }
        swap(nodes[idx], nodes[largest]);
        idx = largest;
    }
  }

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

  void build_from_array(vector<int>temp)
  {
    nodes = temp;
    int n = nodes.size() - 1;
    int last_non_leaf = n/2 - 1;
    for(int i=last_non_leaf; i>=0; i--){
        down_heapify(i);
    }
  }

  int getMax()
  {
    return nodes[0];
  }
};

class MinHeap{
public:
  MaxHeap mx;
```

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

    int getMin()
    {
        return -mx.getMax();
    }

};


int main()
{
    MinHeap mp;
    mp.insert(100);
    mp.insert(150);
    mp.insert(50);
    mp.insert(90);
    cout<<mp.getMin()<<endl;

    mp.Delete(0);
    cout<<mp.getMin()<<endl;

    mp.Delete(0);
    cout<<mp.getMin()<<endl;

    mp.Delete(0);
    cout<<mp.getMin()<<endl;

    return 0;
}
```

Question No. 09

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.

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

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

int main()
{
   int t;cin>>t;
   map<string, int>mp;
   for(int i=0; i<t; i++)
   {
      string temp;
      cin>>temp;
      if(mp.count(temp))
      {
         cout<<mp[temp]<<endl;
      }
      else
      {
         cout<<-1<<endl;
      }
      mp[temp] = i;
```

```
    }

    return 0;
}
```

Question No. 10

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

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

Answer No. 10

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

int main()
{
    int t;cin>>t;
    set<int>st;
    int temp;
    for(int i = 0; i<t; i++){
        cin>>temp;
        st.insert(temp);
    }

    cin>>t;
    for(int i=0; i<t; i++){
        cin>>temp;
        st.insert(temp);
```

```
    }

    for(auto it = st.begin(); it!= st.end(); it++){
        cout<<*it<<" ";
    }

    return 0;
}
```