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>a)
{
   if(a.size()<=1)
      return a;
   int mid = a.size()/2;
   vector<string>b;
   vector<string>c;
   for(int i =0;i<mid;i++)
      b.push_back(a[i]);
   for(int i =mid;i<a.size();i++)
      c.push_back(a[i]);
   vector<string>sorted_b = merge_sort(b);
   vector<string>sorted_c = merge_sort(c);
   vector<string>sorted_a;
   int idx1=0;
   int idx2=0;
   for(int i=0;i<a.size();i++)
   {
      if(idx1==sorted_b.size())
      {
         sorted_a.push_back(sorted_c[idx2]);
         idx2++;
      }
      else if(idx2==sorted_c.size())
```

```
    {
        sorted_a.push_back(sorted_b[idx1]);
        idx1++;
    }
    else if(sorted_b[idx1]<sorted_c[idx2])
    {
        sorted_a.push_back(sorted_b[idx1]);
        idx1++;
    }
    else
    {
        sorted_a.push_back(sorted_c[idx2]);
        idx2++;
    }
  }
  return sorted_a;
}
int main()
{
  int n;
  cin>>n;
  vector<string>a(n);
  for(int i =0;i<n;i++)
  {
    cin>>a[i];
  }
  vector<string>ans=merge_sort(a);
  for(int i=0;i<ans.size();i++)
    cout<<ans[i]<<" ";
  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:
   int data;
   node *prev,*next;
};
class d_linkedlist
{
public:
   node *head,*tail;
   int sz;
   d_linkedlist()
   {
      head=NULL;
      tail=NULL;
      sz=0;
   }
   node *create_new_node(int value)
   {
      node *temp = new node;
      temp->data = value;
      temp->prev = NULL;
      temp->next = NULL;
      return temp;
   }
   void insert_at_head(int value)
   {
      sz++;
      node *a = create_new_node(value);
      if(head==NULL)
      {
         head = a;
         tail=a;
         return ;
      }
      a->next = head;
```

```
    head->prev = a;
    head=a;
  }
  void insert_at_tail(int value)
  {
    sz++;
    node *a = create_new_node(value);
    if(head==NULL)
    {
      head = a;
      tail=a;
      return ;
    }
    tail->next = a;
    a->prev = tail;
    tail=a;
  }

  void insert_at_mid(int value)
  {
    int mid = sz/2;
    node *a = create_new_node(value);
    if(head==NULL)
    {
      head = a;
      tail=a;
      return ;
    }
    node *temp=head;
    while(mid--)
    {
      temp=temp->next;
    }
    node *temp1=temp->prev;
    temp1->next = a;
    a->prev =temp1;
    a->next=temp;
    temp->prev =a;
    sz++;
  }
};
int main()
{
```

```
    d_linkedlist a;

    a.insert_at_head(1);
    a.insert_at_tail(5);
    a.insert_at_mid(3);
    a.insert_at_head(0);
    a.insert_at_tail(10);

    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).

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

Answer No. 03

```
#include<bits/stdc++.h>
using namespace std;
class node
{
public:
    int data;
    node *prev,*next;
};
class d_linkedlist
{
public:
    node *head,*tail;
    int sz;
```

```
d_linkedlist()
{
   head=NULL;
   tail=NULL;
   sz=0;
}
node *create_new_node(int value)
{
   node *temp = new node;
   temp->data = value;
   temp->prev = NULL;
   temp->next = NULL;
   return temp;
}
void insert_at_head(int value)
{
   sz++;
   node *a = create_new_node(value);
   if(head==NULL)
   {
      head = a;
      tail=a;
      return ;
   }
   a->next = head;
   head->prev = a;
   head=a;
}
void insert_at_tail(int value)
{
   sz++;
   node *a = create_new_node(value);
   if(head==NULL)
   {
      head = a;
      tail=a;
      return ;
   }
   tail->next = a;
   a->prev = tail;
   tail=a;
}
```

```
  void insert_at_mid(int value)
  {
     int mid = sz/2;
     node *a = create_new_node(value);
     if(head==NULL)
     {
        head = a;
        tail=a;
        return ;
     }
     node *temp=head;
     while(mid--)
     {
        temp=temp->next;
     }
     node *temp1=temp->prev;
     temp1->next = a;
     a->prev =temp1;
     a->next=temp;
     temp->prev =a;
     sz++;
  }
  void print()
  {
     node *a = head;
     while(a!=NULL)
     {
        cout<<a->data<<" ";
        a=a->next;
     }
     cout<<"\n";
  }
  void Merge(d_linkedlist b)
  {
     tail->next = b.head;
  }
};
int main()
{
  d_linkedlist a;
        d_linkedlist b;

  a.insert_at_head(1);
```

```
    a.insert_at_tail(5);
    a.insert_at_mid(3);
    a.insert_at_head(0);
    a.insert_at_tail(10);
    a.print(); // prints  0 1 3 5 10

    b.insert_at_head(10);
            b.insert_at_tail(50);
            b.insert_at_mid(30);
            b.insert_at_head(9);
            b.insert_at_tail(100);
            b.print(); // prints  0 1 3 5 10

            a.Merge(b);
            a.print();
            b.print();


    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()
{
   string T;
   cin>>T;
   stack<char>s;
   for(int i=0;i<T.size();i++)
   {
      char ch = T[i];
      if(ch=='('||ch=='{'||ch=='[')
         s.push(ch);
      else
      {
         if(s.empty())
         {
            cout<<"No\n";
            return 0;
         }
         if(ch==')'&&s.top()=='(')
            s.pop();
         else if(ch=='}'&&s.top()=='{')
            s.pop();
         else if(ch==']'&&s.top()=='[')
            s.pop();
```

```
        else
        {
           cout<<"No\n";
           return 0;
        }
     }
  }
  if(s.empty())
     cout<<"Yes\n";
  else
     cout<<"No\n";
  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 N= 100;

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

   Queue()
```

```
  {
    l = 0;
    r = -1;
  }
  void Enqueue(int value)
  {
    if(r+1 >= N)
    {
      cout<<"Queue is full\n";
      return;
    }
    r++;
    a[r] = value;
  }
  void Dequeue()
  {
    if(l > r)
    {
      cout<<"Queue is empty\n";
      return;
    }
    l++;
  }
  int Front()
  {
    if(l>r)
    {
      cout<<"Queue is empty\n";
      return -1;
    }
    return a[l];
  }
};

int main()
{
  Queue q;
  q.Enqueue(15);
  q.Enqueue(156);
  q.Enqueue(87);

  cout<<q.Front()<<"\n";
  q.Dequeue();
```

```
   cout<<q.Front()<<"\n";
   q.Dequeue();
   cout<<q.Front()<<"\n";
   q.Dequeue();

   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()
{
   deque<int>d;
   int n;
   cin>>n;
   for(int i =0;i<n;i++)
   {
      int a;
```

```
    cin>>a;
    d.push_back(a);
  }
  vector<int>q;
  for(int i =0;i<n;i++)
  {
    int b =d.front();
    int c = d.back();
    if(b>c)
    {
      q.push_back(c);
      d.pop_back();
    }
    else
    {
      q.push_back(b);
      d.pop_front();
    }

  }
  for(int i =0;i<q.size();i++)
  {
    cout<<q[i]<<" ";
  }
  return 0;
}
```

Question No. 07

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.

Answer No. 07

```
#include<bits/stdc++.h>
using namespace std;
class node
{
public:
   int data;
   node* left;
   node* right;
};
class BST
{
public:
   node *root;
   BST()
   {
      root=NULL;
   }
   node* CreateNewNode(int value)
   {
      node *newnode = new node;
      newnode->data=value;
      newnode->left=NULL;
      newnode->right=NULL;
      return newnode;
   }
   void Insert(int value)
   {
      node *temp=CreateNewNode(value);
      if(root==NULL)
      {
         root=temp;
         return;
      }
      node *prv=NULL;
      node *cur = root;
      while(cur!=NULL)
      {
         if(cur->data>=temp->data)
         {
            prv=cur;
            cur=cur->left;
         }
         else
```

```
        {
            prv=cur;
            cur=cur->right;
        }
    }
    if(prv->data>=temp->data)
    {
        prv->left=temp;
    }
    else
        prv->right=temp;
  }
  bool Search(int value)
  {
    node *temp=root;
    if(root==NULL)
        return false;
    while(temp!=NULL)
    {
        if(temp->data>value)
        {
            temp=temp->left;
        }
        else if(temp->data<value)
        {
            temp=temp->right;
        }
        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";
    cout<<bst.Search(9)<<"\n";
    cout<<bst.Search(20)<<"\n";
    cout<<bst.Search(60)<<"\n";
    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.

Answer No. 08

```
#include<bits/stdc++.h>
using namespace std;
class MaxHeap
{
public:
    vector<int>node;

    MaxHeap()
    {

    }
    void upheapify(int idx)
    {
        while(idx>0&&node[idx]>node[(idx-1)/2])
        {
            swap(node[idx],node[(idx-1)/2]);
            idx=(idx-1)/2;
        }
    }
```

```
void Insert(int x)
{
   node.push_back(x);
   downheapify(node.size()-1);
}
void downheapify(int idx)
{
   while(true)
   {
      int largest = idx;
      int l = 2*idx +1;
      int r = 2*idx +2;
      if(l<node.size() && node[l]>node[largest])
         largest = l;
      if(r<node.size() && node[r]>node[largest])
         largest = r;
      if(largest == idx)
         break;
      swap(node[idx],node[largest]);
      idx = largest;
   }
}
void Delete(int idx)
{
   if(idx>=node.size())
      return;
   swap(node[idx],node[node.size()-1]);
   node.pop_back();
   upheapify(idx);
}
int getMax()
{
   if(node.size()==0)
      return -1;
   return node[0];
}
int ExtractMax()
{
   if(node.size()==0)
      return -1;
   int m = node[0];
   Delete(0);
   return m;
```

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

};
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 mh;
    mh.Insert(10);
    mh.Insert(9);
    mh.Insert(15);
    mh.Insert(8);
    mh.Insert(5);
    mh.Insert(8);
    mh.Insert(5);
    mh.Insert(2);
    mh.Insert(3);
    cout<<mh.getMin()<<"\n";
}
```

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 n;
   cin>>n;
   map<string,int>mp1;
   for(int i =1;i<=n;i++)
   {
      string T;
      cin>>T;
      if(mp1[T]==0)
      {
         cout<<-1<<"\n";
         mp1[T] = i;
         continue;
      }
      cout<<--mp1[T]<<"\n";
      mp1[T]=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

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

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int n;
   cin>>n;
   set<int>s;
   for(int i =0; i<n; i++)
   {
      int a;
      cin>>a;
      s.insert(a);
   }
   int n1;
```

```
    cin>>n1;
    for(int i =0; i<n1; i++)
    {
        int b;
        cin>>b;
        s.insert(b);
    }
    for(auto i:s)
        cout<<i<<" ";
    return 0;
}
```