Answer to the question no-01

```
#include<bits/stdc++.h>
using namespace std;
vector<string>Merge(vector<string>A)
{
   if(A.size()==1)
   {
      return A;
   }
   int m=A.size()/2;
   vector<string>B,C;
   for(int i=0; i<m; i++)
   {
      B.push_back(A[i]);
   }
   for(int i=m; i<A.size(); i++)
   {
      C.push_back(A[i]);
   }
   vector<string>Sort_B=Merge(B);
   vector<string>Sort_C=Merge(C);
   vector<string>Sort_A;
   int id1=0,id2=0,s=A.size();
   for(int i=0; i<s; i++)
   {
      if(id1==Sort_B.size())
      {
         Sort_A.push_back(Sort_C[id2]);
         id2++;
      }
      else if(id2==Sort_C.size())
      {
         Sort_A.push_back(Sort_B[id1]);
         id1++;
      }
      else if(Sort_B[id1]<Sort_C[id2])
      {
            Sort_A.push_back(Sort_B[id1]);
            id1++;
        }
        else
        {
            Sort_A.push_back(Sort_C[id2]);
            id2++;
        }
    }
    return Sort_A;
}
int main()
{
    int n;
    cin>>n;
    getchar();
    vector<string>b(n);
    for(int i=0; i<n; i++)
    {
        cin>>b[i];
    }
    vector<string>A=Merge(b);
    for(int i=0;i<n;i++)
    {
        cout<<A[i]<<" ";
    }
}
```

Answer to the question no-02

```
#include<bits/stdc++.h>
using namespace std;
class node
{
public:
    int data;
    node *nxt;
    node *prv;
```

```
};
class Doubly_Linked_List
{
public:
   node *head;
   node *tail;
   int sz;
   Doubly_Linked_List()
   {
      head = NULL;
      sz = 0;
      tail=NULL;
   }
   node * Create_New_Node(int data)
   {
      node *newnode = new node;
      newnode->data = data;
      newnode->nxt = NULL;
      newnode->prv = NULL;
      return newnode;
   }
   void insertHead(int data)
   {
      sz++;
      node *newnode = Create_New_Node(data);
      if(head == NULL)
      {
         head = newnode;
         tail=newnode;
         return;
      }
      node *a = head;
      newnode->nxt = a;
      a->prv = newnode;
      head = newnode;
   }
   void insertTail(int value)
```

```
{
   if(tail==NULL)
   {
      return;
   }
   node *newnode=Create_New_Node(value);
   tail->nxt=newnode;
   newnode->prv=tail;
   tail=newnode;
   sz++;
}
void insertMid(int value)
{
   Insert(sz/2,value);
}
void Insert(int index, int data)
{
   if(index > sz)
   {
      return;
   }
   if(index==0)
   {
      insertHead(data);
      return;
   }
   node *a = head;
   int cur_index = 0;
   while(cur_index!= index-1)
   {
      a = a->nxt;
      cur_index++;
   }
   node *newnode = Create_New_Node(data);
   newnode->nxt = a->nxt;
   newnode->prv = a;
   node *b = a->nxt;
```

```
        b->prv = newnode;
        a->nxt = newnode;
        sz++;
    }
    void Traverse()
    {
        node *a = head;
        while(a!=NULL)
        {
            cout<<a->data<<" ";
            a = a->nxt;
        }
        cout<<"\n";
    }
    int get_Size()
    {
        return sz;
    }
};
int main()
{
    Doubly_Linked_List dl;
    int a;
    cin>>a;
    for(int i=1;i<=a;i++)
    {
        int b;cin>>b;
        dl.insertHead(b);
    }
    dl.Traverse();
    dl.insertTail(6);
    dl.Traverse();
    dl.insertMid(9);
    dl.Traverse();
    return 0;
}
```

Answer to the question no-03

```
#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;
        tail=NULL;
    }
    node * Create_New_Node(int data)
    {
        node *newnode = new node;
        newnode->value = data;
        newnode->nxt = NULL;
        newnode->prv = NULL;
        return newnode;
    }
    void insertHead(int data)
    {
        sz++;
        node *newnode = Create_New_Node(data);
        if(head == NULL)
        {
```

```
        head = newnode;
        tail=newnode;
        return;
    }
    node *a = head;
    newnode->nxt = a;
    a->prv = newnode;
    head = newnode;
}
void insertTail(int value)
{
    if(tail==NULL)
    {
        return;
    }
    node *newnode=Create_New_Node(value);
    tail->nxt=newnode;
    newnode->prv=tail;
    tail=newnode;
    sz++;
}
void insertMid(int value)
{
    Insert(sz/2,value);
}
void Insert(int index, int data)
{
    if(index > sz)
    {
        return;
    }
    if(index==0)
    {
        insertHead(data);
        return;
    }
    node *a = head;
```

```
        int cur_index = 0;
        while(cur_index!= index-1)
        {
            a = a->nxt;
            cur_index++;
        }
        node *newnode = Create_New_Node(data);
        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 b)
    {
        node *a=b.head;
        while(a!=NULL)
        {
            insertTail(a->value);
            a=a->nxt;
        }
    }
};
int main()
{
    LinkedList a;
```

```
        LinkedList b;

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

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

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

}
```

<u>**Answer to the question no-04**</u>

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   stack<char>S;
   string s;
   cin>>s;
   for(int i=0;i<s.size();i++)
   {
      if(s[i]=='('||s[i]=='{'||s[i]=='[')
      {
         S.push(s[i]);
      }
      else
```

```
        {
           if(s[i]==')'&&S.top()=='(')
           {
              S.pop();
           }
           else if(s[i]=='}'&&S.top()=='{')
           {
              S.pop();
           }
           else if(s[i]==']'&&S.top()=='[')
           {
              S.pop();
           }
           else
           {
              cout<<"NO\n";
              return 0;
           }
        }
    }
    S.size()==0?cout<<"YES\n":cout<<"NO\n";
}
```

<u>Answer to the question no-05</u>

```
#include<bits/stdc++.h>
using namespace std;
template< class Adil>
class Queue
{
public:
    int MAX=100;
    Adil A[100];
    int l,r,sz;

    Queue()
    {
        sz=0;l=0;r=-1;
    }
    void enqueue(Adil value)
    {
        if(sz==MAX)
        {
            cout<<"Queue is full";
            return;
        }
        r++;
        if(r==MAX)
        {
            r=0;
        }
        A[r]=value;
        sz++;
    }
    void dequeue()
    {
        if(sz==0)
        {
            cout<<"Queue is empty";
            return;
        }
        l++;
        if(l==MAX)
        {
            l=0;
        }
        sz--;
    }
    Adil Front()
    {
        return A[l];
    }
};
int main()
{
    Queue<int>Q;
    int n;cin>>n;
    for(int i=1;i<=n;i++)
    {
        int a;cin>>a;
        Q.enqueue(a);
    }
    cout<<Q.Front()<<" ";
    Q.dequeue();
    cout<<Q.Front()<<" ";
}
```

<u>Answer to the question no-06</u>

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    deque<int>d1,d2;
    int n;cin>>n;
    for(int i=1;i<=n;i++)
    {
        int a;cin>>a;
        d1.push_back(a);
    }
    while(d1.size()!=0)
    {
        int a=d1.front(),b=d1.back();
        if(a<b)
        {
            d2.push_back(a);
            d1.pop_front();
        }
        else if(a>b)
        {
            d2.push_back(b);
            d1.pop_back();
        }
        else
        {
           d2.push_back(a);
           d2.push_back(b);
           d1.pop_back();d1.pop_front();
        }
    }
}
```

<u>Answer to the question no-07</u>

```
#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 *Create_new_node(int value)
    {
       node *newnode= new node;
       newnode->value=value;
       newnode->Left=NULL;
       newnode->Right=NULL;
       return newnode;
    }
    void Insert(int value)
    {
       node *newnode=Create_new_node(value);
       if(root==NULL)
    {
        root=newnode;
        return;
    }
    queue<node*>Q;
    Q.push(root);
    while(Q.size()!=0)
    {
        node *a=Q.front();
        Q.pop();
        if(a->Left!=NULL)
        {
            Q.push(a->Left);
        }
        else
        {
            a->Left=newnode;
            return;
        }
        if(a->Right!=NULL)
        {
            Q.push(a->Right);
        }
        else
        {
            a->Right=newnode;
            return;
        }
    }
}
bool Search(int value)
{
    int a=Searching(value);
    if(a==1)
    {
        return true;
    }
```

```
        return false;
    }
private:
    int Searching(int value)
    {
        queue<node*>Q;
        Q.push(root);
        while(!Q.empty())
        {
            node *a=Q.front();
            Q.pop();
            if(a->Left!=NULL)
            {
                Q.push(a->Left);
            }
            if(a->Right!=NULL)
            {
                Q.push(a->Right);
            }
            if(a->value==value)
            {
                return 1;
            }
        }
        return 0;
    }
};
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 to the question no-08

```
class MinHeap
{
public:
    Max_Heap<int>mx;
    void insert(int x)
    {
        mx.Insert(-x);
    }
    void Delete(int idx)
    {
        mx.Delete(idx);
    }
    int getMin()
    {
        return -mx.get_max();
    }
};
```

Answer to the question no-09

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    map<string,int>mp;
    int n;cin>>n;
    getchar();
    for(int i=1;i<=n;i++)
    {
        string s;
        cin>>s;
        if(mp[s]==0)
        {
           cout<<-1<<"\n";
           mp[s]=i;
        }
        else
        {
           cout<<mp[s]-1<<"\n";
           mp[s]=i;
        }
    }
}
```

Answer to the question no-10

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    set<int>S;
    int a,b;cin>>a;
    for(int i=1;i<=a;i++)
    {
        int n;cin>>n;
        S.insert(n);
    }
    cin>>b;
    for(int i=1;i<=b;i++)
    {
        int n;cin>>n;
        S.insert(n);
    }
    for(auto it:S)
    {
        cout<<it<<" ";
    }
}
```