Answer Script

Question No. 01

1. Write a program to reverse an array. 10

Sample input	Sample output
5 6 2 3 3 5	5 3 3 2 6

Answer No. 01

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int n;
   cin>>n;
   int A[n];
   for(int i=n-1;i>=0;i--)
   {
      cin>>A[i];
   }
   for(int i=0;i<n;i++)
   {
      cout<<A[i]<<" ";
   }
}
```

Question No. 02

1. Write a program to remove duplicate numbers from an array and print the remaining elements in sorted order. You have to do this in O(nlogn). 15

Sample input	Sample output
5 6 3 2 3 5	2 3 5 6

Answer No. 02

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

```
{
   int n;
   cin>>n;
   int A[n];
   vector<int>B(1000);
   for(int i=0;i<n;i++)
   {
      cin>>A[i];
      B[A[i]]++;
   }
   vector<int>C;
   for(int i=0;i<n;i++)
   {
      if(B[A[i]]>0)
      {
         C.push_back(A[i]);
         B[A[i]]=0;
      }
   }
   sort(C.begin(),C.end());
   for(int i=0;i<C.size();i++)
   {
      cout<<C[i]<<" ";
   }

}
```

Question No. 03

1. Write a program to sort the numbers in non-increasing order using quick sort. You have to take random index as a pivot element. 15

Sample input	**Sample output**
5 6 3 2 3 5	6 5 3 3 2

Answer No. 03

```
#include<bits/stdc++.h>
using namespace std;
vector<int>quick_sort(vector<int>A)
{
```

```
    if(A.size()<=1){return A;}
    int P=rand()%(A.size());
    vector<int>B,C;
    for(int i=0;i<A.size();i++)
    {
        if(i==P){continue;}
        if(A[i]<=A[P]){B.push_back(A[i]);}
        else{C.push_back(A[i]);}
    }
    vector<int>Sort_B=quick_sort(B);
    vector<int>Sort_C=quick_sort(C);
    vector<int>Sort_A;
    for(int i=0;i<Sort_C.size();i++)
    {
        Sort_A.push_back(Sort_C[i]);
    }
    Sort_A.push_back(A[P]);
    for(int i=0;i<Sort_B.size();i++)
    {
        Sort_A.push_back(Sort_B[i]);
    }
    return Sort_A;
}

int main()
{
   int n;
   cin>>n;
   vector<int>a(n);
   for(int i=0;i<n;i++)
   {
      cin>>a[i];
   }
   vector<int>Sort_A=quick_sort(a);
   for(int i=0;i<Sort_A.size();i++)
   {
      cout<<Sort_A[i]<<" ";
   }
   cout<<"\n";
}
```

Question No. 04

1. Write a recursive function to check if a given word is a palindrome.
   15

Sample input	**Sample output**
abcba	Yes
abcaa	No

A palindrome is a word which reads the same forward and backward.

Answer No. 04

```
#include<bits/stdc++.h>
using namespace std;
int palindrome(string s, int n, int m)
{
   if(n==m)
   {
      return 1;
   }
   if(m>n)
   {
      return 1;
   }
   if(s[n]!=s[m])
   {
      return 0;
   }
   return palindrome(s,n-1,m+1);
}
int main()
{
  int r;
  string s;
  cin>>s;
  r=s.size()-1;
  int x=palindrome(s,r,0);
  if(x==1)
  {
     cout<<"YES";
  }
  else
  {
     cout<<"NO";
```

```
  }
}
```

Question No. 05

1. Write a recursive function to find the maximum element in an array.
   15

Sample input	**Sample output**
5 1 3 5 2 4	5

Answer No. 05

```
#include<bits/stdc++.h>
using namespace std;
int maxx(int A[],int n)
{
   if(n==0)
   {
      return A[n-1];
   }
   int maxi=maxx(A,n-1);
   if(maxi>A[n-1])
   {
      return maxi;
   }
   else
   {
      return A[n-1];
   }
}
int main()
{
  int n;
  cin>>n;
  int A[n];
  for(int i=0;i<n;i++)
  {
     cin>>A[i];
  }
  cout<<maxx(A,n);
}
```

Question No. 06

Add the following functions to the class.

- **int getLast()** -> This function will return the last node of the linked list. If the linked list is empty then return -1.
  Sample Input: [3, 2, 6, 4, 5]
  Sample Output: 5
- **double getAverage()** -> This function will return the average of all elements in the linked list.
  Sample Input: [3, 2, 6, 4, 7]
  Sample Output: 4.4

Answer No. 06

```
#include<bits/stdc++.h>
using namespace std;
class node
{
public:
   int data;
   node * nxt;
};
class LinkedList
{
public:
   node * head;
   int sz;
   LinkedList()
   {
      head = NULL;
      sz=0;
   }
   node* CreateNewNode(int value)
   {
      node *newnode = new node;
      newnode->data = value;
      newnode->nxt = NULL;
      return newnode;
   }
```

```
void InsertAtHead(int value)
{
   sz++;
   node *a = CreateNewNode(value);
   if(head == NULL)
   {
      head = a;
      return;
   }
   a->nxt = head;
   head = a;
}
void Traverse()
{
   node* a = head;
   while(a!= NULL)
   {
      cout<<a->data<<" ";
      a = a->nxt;
   }
   cout<<"\n";
}
int SearchDistinctValue(int value)
{
   node* a = head;
   int index = 0;
   while(a!= NULL)
   {
      if(a->data==value)
      {
         return index;
      }
      a = a->nxt;
      index++;
   }
   return -1;
}
void SearchAllValue(int value)
{
   node* a = head;
   int index = 0;
   while(a!= NULL)
   {
```

```
        if(a->data==value)
        {
           cout<<value<<" is found at index "<<index<<"\n";
        }
        a = a->nxt;
        index++;
    }
}
int getSize()
{
    return sz;
}
void InsertAtAnyIndex(int index, int value)
{
    if(index <0 || index > sz)
    {
        return;
    }
    if(index==0)
    {
        InsertAtHead(value);
        return;
    }
    sz++;
    node *a = head;
    int cur_index = 0;
    while(cur_index!=index-1)
    {
        a = a->nxt;
        cur_index++;
    }
    node *newnode = CreateNewNode(value);
    newnode->nxt = a->nxt;
    a->nxt = newnode;
}
void DeleteAtHead()
{
    if(head == NULL)
    {
        return;
    }
    sz--;
    node *a = head;
```

```
    head = a->nxt;
    delete a;
}
void DeleteAnyIndex(int index)
{
    if(index <0 || index > sz-1)
    {
        return;
    }
    if(index==0)
    {
        DeleteAtHead();
        return;
    }
    sz--;
    node *a = head;
    int cur_index = 0;
    while(cur_index != index-1)
    {
        a = a->nxt;
        cur_index++;
    }
    node *b = a->nxt;
    a->nxt = b->nxt;
    delete b;
}

void InsertAfterValue(int value , int data)
{
    node *a = head;
    while(a != NULL)
    {
        if(a->data == value)
        {
            break;
        }
        a = a->nxt;
    }
    if(a== NULL)
    {
        cout<<value<<" doesn't exist in linked-list.\n";
        return;
    }
```

```
    sz++;
    node *newnode = CreateNewNode(data);
    newnode->nxt = a->nxt;
    a->nxt = newnode;
  }
  void ReversePrint2(node *a)
  {
    if(a==NULL)
    {
      return;
    }
    ReversePrint2(a->nxt);
    cout<<a->data<<" ";
  }
  void ReversePrint()
  {
    ReversePrint2(head);
    cout<<"\n";
  }
  int getLast()
  {
    if(head==NULL)
    {
      return -1;
    }
    node *a=head;
    return a->data;
  }
  double getAverage()
  {
    node *a=head;
    double b=0;
    while(a!=NULL)
    {
      b+=(double)a->data;
      a=a->nxt;
    }
    return b/sz;
  }
};
int main()
{
  LinkedList l;
```

```
    for(int i=1;i<=5;i++)
    {
        int a;
        cin>>a;
        l.InsertAtHead(a);
    }
    cout<<l.getLast()<<"\n";
    printf("%.1lf\n",l.getAverage());
    return 0;
}
```

Question No. 07

Add the following functions to the class.

- **void swap(i , j)** -> This function will swap the i-th index and j-th index.
  Sample Input: [3, 2, 6, 4, 7], i = 1, j = 4
  Sample Output: Doubly Linked list containing the elements [3,7,6,4,2]
- **void deleteZero()** -> This function will delete all the nodes that have data=0.
  Sample Input: [0, 2, 0, 0, 5]
  Sample Output: Doubly linked list containing the elements [2, 5]

Answer No. 07

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

```
    head = NULL;
    sz = 0;
}
node * CreateNewNode(int data)
{
    node *newnode = new node;
    newnode->data = data;
    newnode->nxt = NULL;
    newnode->prv = NULL;
    return newnode;
}
void InsertAtHead(int data)
{
    sz++;
    node *newnode = CreateNewNode(data);
    if(head == NULL)
    {
        head = newnode;
        return;
    }
    node *a = head;
    newnode->nxt = a;
    a->prv = newnode;
    head = newnode;
}
void Insert(int index, int data)
{
    if(index > sz)
    {
        return;
    }
    if(index==0)
    {
        InsertAtHead(data);
        return;
    }
    node *a = head;
    int cur_index = 0;
    while(cur_index!= index-1)
    {
        a = a->nxt;
        cur_index++;
    }
```

```
    node *newnode = CreateNewNode(data);
    newnode->nxt = a->nxt;
    newnode->prv = a;
    node *b = a->nxt;
    b->prv = newnode;
    a->nxt = newnode;
    sz++;
}
void Delete(int index)
{
    node *a = head;
    int cur_index = 0;
    while(cur_index != index)
    {
        a = a->nxt;
        cur_index++;
    }
    node *b = a->prv;
    node *c = a->nxt;
    if(b!=NULL)
    {
        b->nxt = c;
    }
    if(c!= NULL)
    {
        c->prv = b;
    }
    delete a;
    if(index==0)
    {
        head = c;
    }
    sz--;
}
void Traverse()
{
    node *a = head;
    while(a!=NULL)
    {
        cout<<a->data<<" ";
        a = a->nxt;
    }
    cout<<"\n";
```

```
}
int getSize()
{
   return sz;
}
void Reverse()
{
   if(head==NULL)
   {
      return;
   }
   node *a = head;
   int cur_index = 0;
   while(cur_index != sz-1)
   {
      a = a->nxt;
      cur_index++;
   }
   node *b = head;
   while(b!= NULL)
   {
      swap(b->nxt, b->prv);
      b = b->prv;
   }
   head = a;
}
void Swap(int i,int j)
{
   if(i>sz||j>sz||i==j)
   {
      return;
   }
   Reverse();
   node *a=head;
   node *b,*c;
   int d=-1;
   while(a!=NULL)
   {
      if(d==i)
      {
         b=a;
      }
      if(d==j)
```

```
        {
            c=a;
        }
        a=a->nxt;
        d++;
    }
    int z;
    z=b->data;
    b->data=c->data;
    c->data=z;
}
void deleteZero()
{
    node *a=head;
    vector<int>A;
    int i=0;
    while(a!=NULL)
    {
        if(a->data!=0)
        {
            A.push_back(a->data);
            i++;
        }
        a=a->nxt;
    }
    if(i>0)
    {
        while(head!=NULL)
        {
            node *a=head;
            head=head->nxt;
            delete a;
            sz--;
        }
        for(int i=0;i<A.size();i++)
        {
            InsertAtHead(A[i]);
        }
    }
    else
    {
        return;
    }
```

```
    }
};
int main()
{
    DoublyLinkedList dl;
    for(int i=1; i<=5; i++)
    {
        int a;
        cin>>a;
        dl.InsertAtHead(a);
    }
    dl.deleteZero();
    dl.Traverse();
    int x,y;
    cin>>x>>y;
    dl.Swap(x,y);
    dl.Traverse();
    return 0;
}
```