Answer Script

Question No. 01

Write a program to reverse an array.

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 t;cin>>t;
   vector<int>a(t);
   for(int i=0; i<a.size(); i++)
      cin>>a[i];
   for(int j=a.size()-1; j>=0; j--)
      cout<<a[j]<<" ";

   return 0;
}
```

Question No. 02

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).

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 t;cin>>t;
   vector<int>x(t);
   for(int i=0; i<t; i++)
      cin>>x[i];

   sort(x.begin(), x.end());

   for(int i=1; i<x.size(); i++){
      if(i == 1) cout<<x[0]<<" ";
      if(x[i-1] != x[i])
         cout<<x[i]<<" ";
   }
   return 0;
}
```

Question No. 03

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

e input		**Sample output**
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 pivot = rand()%(a.size());
   vector<int>left, right;

   for(int i=0; i<a.size(); i++){
      if(pivot==i)
         continue;
      else if(a[i]>=a[pivot])
         left.push_back(a[i]);
      else if(a[i]<a[pivot])
         right.push_back(a[i]);
   }
   vector<int>sorted_left = quick_sort(left);
   vector<int>sorted_right = quick_sort(right);
   vector<int>sorted_a;

   for(int i=0; i<sorted_left.size(); i++)
      sorted_a.push_back(sorted_left[i]);
   sorted_a.push_back(a[pivot]);
   for(int j=0; j<sorted_right.size(); j++)
      sorted_a.push_back(sorted_right[j]);

   return sorted_a;
}



int main(){
```

```
    int t;cin>>t;
    vector<int>x(t);
    for(int i=0; i<t; i++)
        cin>>x[i];
    x = quick_sort(x);
    for(int j : x)
        cout<<j<<" ";
    return 0;
}
```

Question No. 04

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

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

Answer No. 04

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

string rev_str(string x, int l){
    string ans;
    if(l<0)return ans;
    return ans + x[l]+ rev_str(x, l-1);
}
int main(){
    string s;cin>>s;
    string temp = rev_str(s, s.size()-1);
    if(s == temp)
        cout<<"YES"<<endl;
    else
```

```
        cout<<"NO"<<endl;

    return 0;
}
```

Question No. 05

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

Sample input	Sample output
5 1 3 5 2 4	5

Answer No. 05

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

int max_ele(vector<int> x, int t, int maxi){
    if(t<0)return maxi;
    if(maxi < x[t]) maxi = x[t];
    maxi = max_ele(x, t-1, maxi);
    return maxi;
}

int main(){
    int t;cin>>t;
    int maxi = -99999999;
    vector<int>x(t);
    for(int i=0; i<t; i++)
        cin>>x[i];

    cout<<max_ele(x, t-1, maxi)<<endl;
```

```
    return 0;
}
```

Question No. 06
Take the Singly linked-list class from Github. Link: https://github.com/phitronio/Data-Structure-Batch2/blob/main/Week%204/Module%2013/1.cpp 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 getSize(){
        return sz;
    }

    int getLast(){
        if(head == NULL)return -1;
        return head->data;
    }

    double getAverage(){
        double avg;
        node* a = head;
        while(a!= NULL){
            avg += a->data;
            a = a->nxt;
        }
        return avg/getSize();
    }


};

int main()
{
    LinkedList l;
    LinkedList l2;
    vector<int>input = {3, 2, 6, 4, 5};
    vector<int>input2 = {3, 2, 6, 4, 7};
    for(int i : input)
        l.InsertAtHead(i);
    for(int j : input2)
        l2.InsertAtHead(j);
```

```
    cout<<l.getLast()<<endl;
    cout<<l2.getAverage();
    return 0;
}
```

Question No. 07

Take the Doubly linked-list class from Github.
Link:
https://github.com/phitronio/Data-Structure-Batch2/blob/main/Week%204/Module%2014/1.cpp

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;
   node *tail;
   int sz;
   DoublyLinkedList()
   {
      head = NULL;
      tail = 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 Traverse()
{
    node *a = head;
    while(a!=NULL)
    {
        cout<<a->data<<" ";
        a = a->nxt;
    }
    cout<<"\n";
}

int getSize()
{
    return sz;
}
void printReverse(){
    node* a = head;
    reverse(a);
    cout<<"\n";
}

void reverse(node* a){
    if (a == NULL) return;
    reverse(a->nxt);
```

```
    cout<<a->data<<" ";
}


void deleteZero()
{
    node *a = head;
    while (a != NULL)
    {
        if (a->data == 0)
        {
            if (a == head)
            {
                head = a->nxt;
                head->prv = NULL;
            }
            else if (a->nxt == NULL)
            {
                tail = a->prv;
                tail->nxt = NULL;
            }
            else
            {
                a->prv->nxt = a->nxt;
                a->nxt->prv = a->prv;
            }
            node *temp = a;
            a = a->nxt;
            delete temp;
        }
        else
        {
```

```
            a = a->nxt;
        }
    }
    Traverse();
}


void swap(int i, int j){
    i--, j--;
    node *current_i = head;
    node *current_j = head;
    node *prev_i = NULL;
    node *prev_j = NULL;

    int count = 0;
    while (count < i){
        prev_i = current_i;
        current_i = current_i->nxt;
        count++;
    }

    count = 0;
    while (count < j){
        prev_j = current_j;
        current_j = current_j->nxt;
        count++;
    }

    if (prev_i != NULL){
        prev_i->nxt = current_j;
    }
    else{
```

```
            head = current_j;
        }

        if (prev_j != NULL){
            prev_j->nxt = current_i;
        }
        else{
            head = current_i;
        }

        node *temp = current_i->nxt;
        current_i->nxt = current_j->nxt;
        current_j->nxt = temp;

        printReverse();

        }




};

int main()
{
    DoublyLinkedList dl;
    DoublyLinkedList dl2;
    vector<int>x = {0, 2, 0, 0, 5};
    for(int i : x)
        dl.InsertAtHead(i);

    dl.deleteZero();
```

```
    vector<int>y = {3, 2, 6, 4, 7};
    for(int j : y)
        dl2.InsertAtHead(j);

    dl2.swap(1, 4);


    return 0;
}
```