Answer Script

Question No. 01

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, i, j;
        cin >> n;
        vector <int> a(n);
        for(i = 0; i < n; i++) {
        cin >> a[i];
        }
        for(i = 0, j = n-1; i < n && i < j; i++, j--) {
        swap(a[i], a[j]);
        }
        for(i = 0; i < n; i++) {
        cout << a[i] << " ";
        }
        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). 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, i;
        cin >> n;
        vector<int> a(n);
        for (i = 0; i < n; i++) {
        cin >> a[i];
        }

        sort(a.begin(), a.end());
        vector <int> a_new;
        for(i = 0; i < a.size(); i++) {
        if(a[i] != a[i+1]) {
        a_new.push_back(a[i]);
        }
        }

        for(i = 0; i < a_new.size(); i++) {
        cout << a_new[i] << " ";
        }
        cout << "\n";

        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.

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;

int partition(vector<int> &v, int low, int high) {
        int pivotIndex = rand() % (high - low + 1) + low;
        swap(v[pivotIndex], v[high]);
        int pivot = v[high];
        int i = low - 1;
        for (int j = low; j <= high - 1; j++) {
        if (v[j] >= pivot) {
        i++;
        swap(v[i], v[j]);
        }
        }
        swap(v[i + 1], v[high]);
        return i + 1;
}

void quickSort(vector<int> &v, int low, int high) {
        if (low < high) {
        int pivot = partition(v, low, high);
        quickSort(v, low, pivot - 1);
        quickSort(v, pivot + 1, high);
        }
}

int main() {
        int n;
        cin >> n;
        vector<int> v(n);
        for (int i = 0; i < n; i++) {
        cin >> v[i];
        }
        quickSort(v, 0, n - 1);
        for (int i = 0; i < n; i++) {
        cout << v[i] << " ";
        }
        return 0;
}
```

Question No. 04

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;

bool isPalindrome(string word, int start, int end) {
        if (start >= end) {
        return true;
        }
        if (word[start] != word[end]) {
        return false;
        }
        return isPalindrome(word, start + 1, end - 1);
}

int main() {
        string word;
        cin >> word;
        if (isPalindrome(word, 0, word.length() - 1)) {
        cout << "Yes";
        } else {
        cout << "No";
        }
        return 0;
}
```

Question No. 05

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 findMax(int arr[], int n) {
        if (n == 1) return arr[0];
        return max(arr[n-1], findMax(arr, n-1));
}

int main() {
        int n;
        cin >> n;
        int arr[n];
        for (int i = 0; i < n; i++) cin >> arr[i];
        cout << findMax(arr, n) << endl;
        return 0;
}
```

Question No. 06

Take the Singly linked-list class from Github.
15
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;
        }

        //Creates a new node with data = value and nxt= NULL
```

```
node* CreateNewNode(int value)
{
node *newnode = new node;
newnode->data = value;
newnode->nxt = NULL;
return newnode;
}
// gets the last element of the linked list
int getLast()
{
if (head == NULL)
return -1;

node *a = head;
while (a->nxt != NULL)
{
a = a->nxt;
}
return a->data;
}
// gets the average value of linked list
double getAverage()
{
if (head == NULL)
return -1;

double sum = 0;
node *a = head;
while (a != NULL)
{
sum += a->data;
a = a->nxt;
}
return sum / sz;
}

// Insert new value at Head
void InsertAtHead(int value)
{
sz++;
node *a = CreateNewNode(value);
if(head == NULL)
{
        head = a;
        return;
        }
        //If head is not NULL
        a->nxt = head;
        head = a;
        }

        //Prints the linked list
        void Traverse()
        {
        node* a = head;
        while(a!= NULL)
        {
        cout<<a->data<<" ";
        a = a->nxt;
        }
        cout<<"\n";
        }
};

int main()
{
        LinkedList l;

        l.InsertAtHead(7);
        l.InsertAtHead(4);
        l.InsertAtHead(6);
        l.InsertAtHead(2);
        l.InsertAtHead(3);

        cout << "Last Element: " << l.getLast() << "\n";
        cout << "Average: " << l.getAverage() << "\n";


        return 0;
}
```

Question No. 07

Take the Doubly linked-list class from Github.
15
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;
        int sz;
        DoublyLinkedList()
        {
        head = NULL;
        sz = 0;
        }
```

```
//Creates a new node with the given data and returns it O(1)
node * CreateNewNode(int data)
{
node *newnode = new node;
newnode->data = data;
newnode->nxt = NULL;
newnode->prv = NULL;
return newnode;
}

//Inserts a node with given data at head O(1)
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;
}

//Prints the linked list O(n)
void Traverse()
{
node *a = head;
while(a!=NULL)
{
cout<<a->data<<" ";
a = a->nxt;
}
cout<<"\n";
}
// swaps the values of given indexes
void swap(int i, int j)
{
if (i >= sz || j >= sz) {
cout << "Invalid" << endl;
return;
```

```
        }
        if (i == j) return;
        node* a = head;
        node* b = head;
        for (int idx = 0; idx < i; idx++) {
        a = a->nxt;
        }
        for (int idx = 0; idx < j; idx++) {
        b = b->nxt;
        }
        int temp = a->data;
        a->data = b->data;
        b->data = temp;
        }
        // deletes the values that is zero
        void deleteZero()
{
        node *curr = head;
        while (curr)
        {
        if (curr->data == 0)
        {
        if (curr == head)
        {
                head = curr->nxt;
                if(curr->nxt != NULL) curr->nxt->prv = NULL;
        }
        else if (curr->nxt == NULL)
        {
                curr->prv->nxt = NULL;
        }
        else
        {
                curr->prv->nxt = curr->nxt;
                curr->nxt->prv = curr->prv;
        }
        sz--;
        }
        curr = curr->nxt;
        }
        }
};
```

```
int main()
{
        DoublyLinkedList dl;
        dl.InsertAtHead(7);
        dl.InsertAtHead(0);
        dl.InsertAtHead(0);
        dl.InsertAtHead(2);
        dl.InsertAtHead(0);

        cout << "Before swapping: ";
        dl.Traverse();
        dl.swap(1, 4);
        cout << "After swapping: ";
        dl.Traverse();

        dl.deleteZero();
        cout << "After deleting zero values: ";
        dl.Traverse();

        return 0;
}
```