Data Structure Lab

CEN-391

LAB FILE

Name : Vicky Gupta

Roll No. : 20BCS070

Branch : Computer Engineering

Subject : Data Structure Lab

INDEX

Data Structure Lab

CEN-391

Program 1

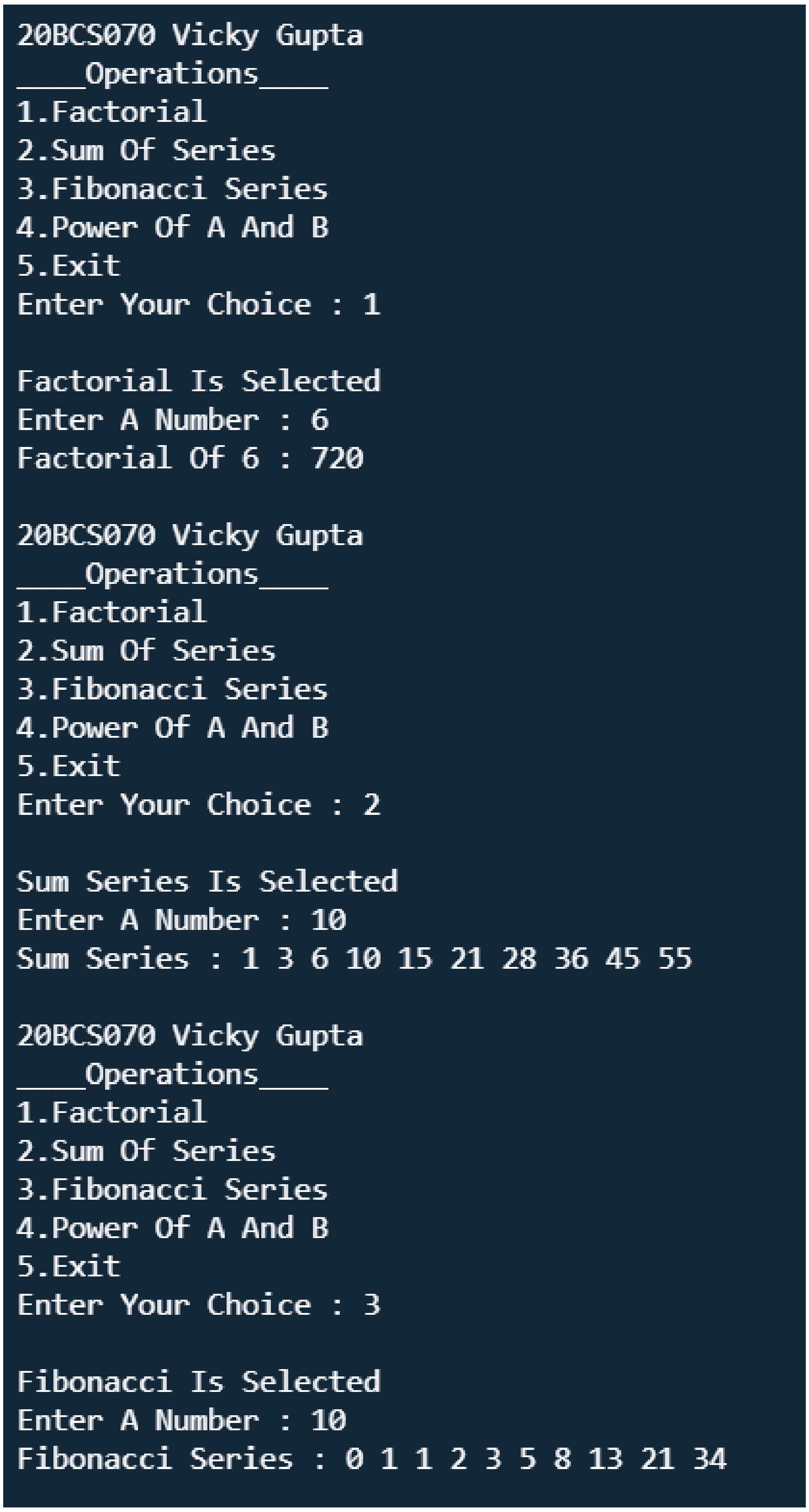
Code :-

```
#include <iostream>
using namespace std;

void Factorial()
{
    cout << endl
         << "Factorial Is Selected" << endl;
    int n, fact = 1;
    cout << "Enter A Number : ";
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        fact *= i;
    }
    cout << "Factorial Of " << n << " : " << fact << endl
         << endl;
}

void Sum_Series()
{
    cout << endl
         << "Sum Series Is Selected" << endl;
    int n;
    cout << "Enter A Number : ";
```

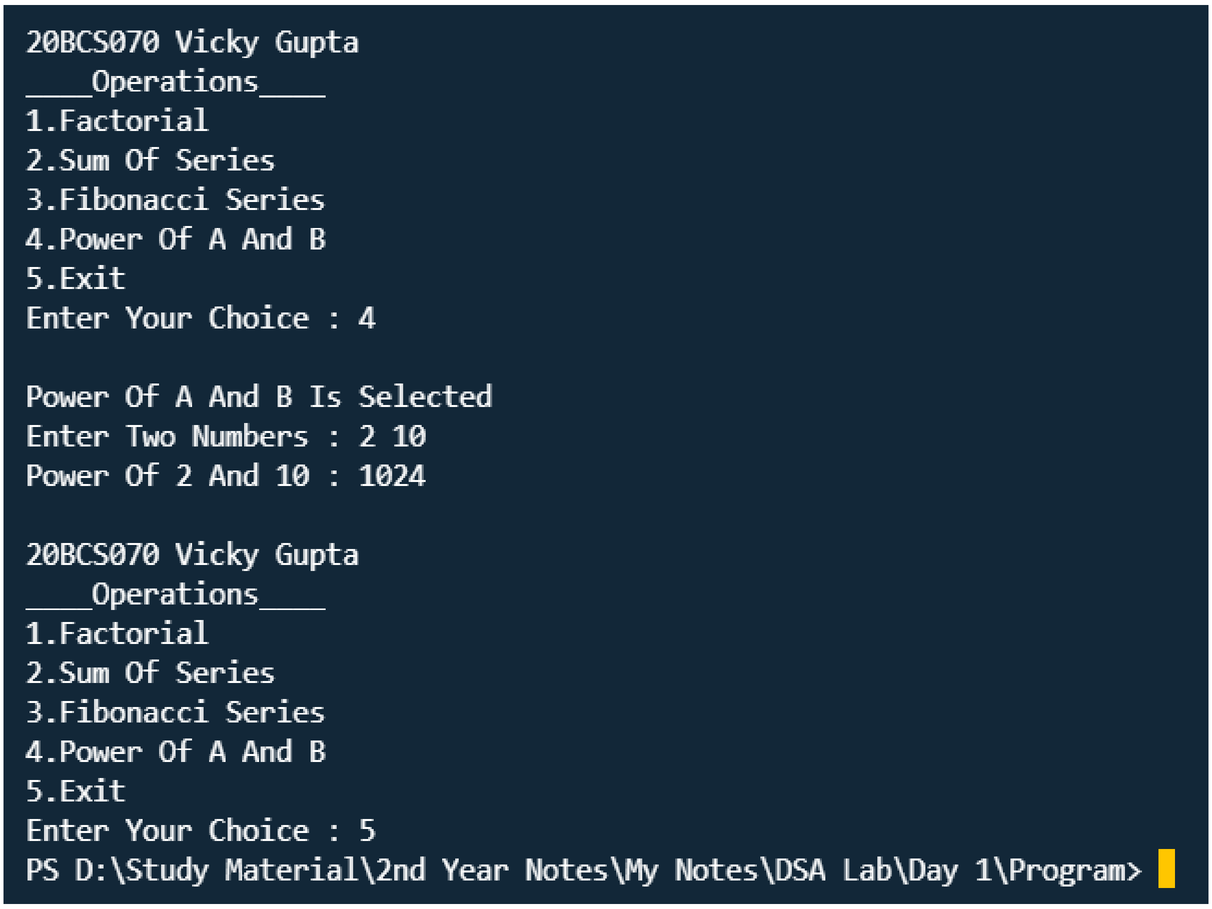
```cpp
    cin >> n;
    int sum = 0;
    cout << "Sum Series : ";
    for (int i = 1; i <= n; i++)
    {
        sum += i;
        cout << sum << " ";
    }
    cout << endl
         << endl;
}

void Fibonacci()
{
    cout << endl
         << "Fibonacci Is Selected" << endl;
    int n;
    cout << "Enter A Number : ";
    cin >> n;
    int num1 = -1, num2 = 1;
    cout << "Fibonacci Series : ";
    for (int i = 0; i < n; i++)
    {
        int num3 = num1 + num2;
        num1 = num2;
        num2 = num3;
        cout << num3 << " ";
    }
    cout << endl
         << endl;
}

void Power()
{
    cout << endl
         << "Power Of A And B Is Selected" << endl;
    int a, b;
    cout << "Enter Two Numbers : ";
    cin >> a >> b;
    int p = 1;
    for (int i = 0; i < b; i++)
    {
        p *= a;
    }
    cout << "Power Of " << a << " And " << b << " : " << p << endl
         << endl;
}

void Menu()
```

```
{
    cout<<"20BCS070 Vicky Gupta"<<endl;
    cout << "____Operations____" << endl;
    cout << "1.Factorial" << endl;
    cout << "2.Sum Of Series" << endl;
    cout << "3.Fibonacci Series" << endl;
    cout << "4.Power Of A And B" << endl;
    cout << "5.Exit" << endl;
    cout << "Enter Your Choice : ";
}

bool Operation()
{
    int n;
    cin >> n;
    switch (n)
    {
    case 1:
        Factorial();
        break;
    case 2:
        Sum_Series();
        break;
    case 3:
        Fibonacci();
        break;
    case 4:
        Power();
        break;
    case 5:
        return false;
    default:
        cout <<endl<< "Invalid Input Try Again!" << endl<<endl;

    }
    return true;
}

int main()
{
    system("cls");
    while (1)
    {
        Menu();
        if (!Operation())
            break;
    }
    return 0;
}
```

Output :-



```
20BCS070 Vicky Gupta
____Operations____
1.Factorial
2.Sum Of Series
3.Fibonacci Series
4.Power Of A And B
5.Exit
Enter Your Choice : 1

Factorial Is Selected
Enter A Number : 6
Factorial Of 6 : 720

20BCS070 Vicky Gupta
____Operations____
1.Factorial
2.Sum Of Series
3.Fibonacci Series
4.Power Of A And B
5.Exit
Enter Your Choice : 2

Sum Series Is Selected
Enter A Number : 10
Sum Series : 1 3 6 10 15 21 28 36 45 55

20BCS070 Vicky Gupta
____Operations____
1.Factorial
2.Sum Of Series
3.Fibonacci Series
4.Power Of A And B
5.Exit
Enter Your Choice : 3

Fibonacci Is Selected
Enter A Number : 10
Fibonacci Series : 0 1 1 2 3 5 8 13 21 34
```



```
20BCS070 Vicky Gupta
____Operations____
1.Factorial
2.Sum Of Series
3.Fibonacci Series
4.Power Of A And B
5.Exit
Enter Your Choice : 4

Power Of A And B Is Selected
Enter Two Numbers : 2 10
Power Of 2 And 10 : 1024

20BCS070 Vicky Gupta
____Operations____
1.Factorial
2.Sum Of Series
3.Fibonacci Series
4.Power Of A And B
5.Exit
Enter Your Choice : 5
PS D:\Study Material\2nd Year Notes\My Notes\DSA Lab\Day 1\Program>
```

Data Structure Lab

CEN-391

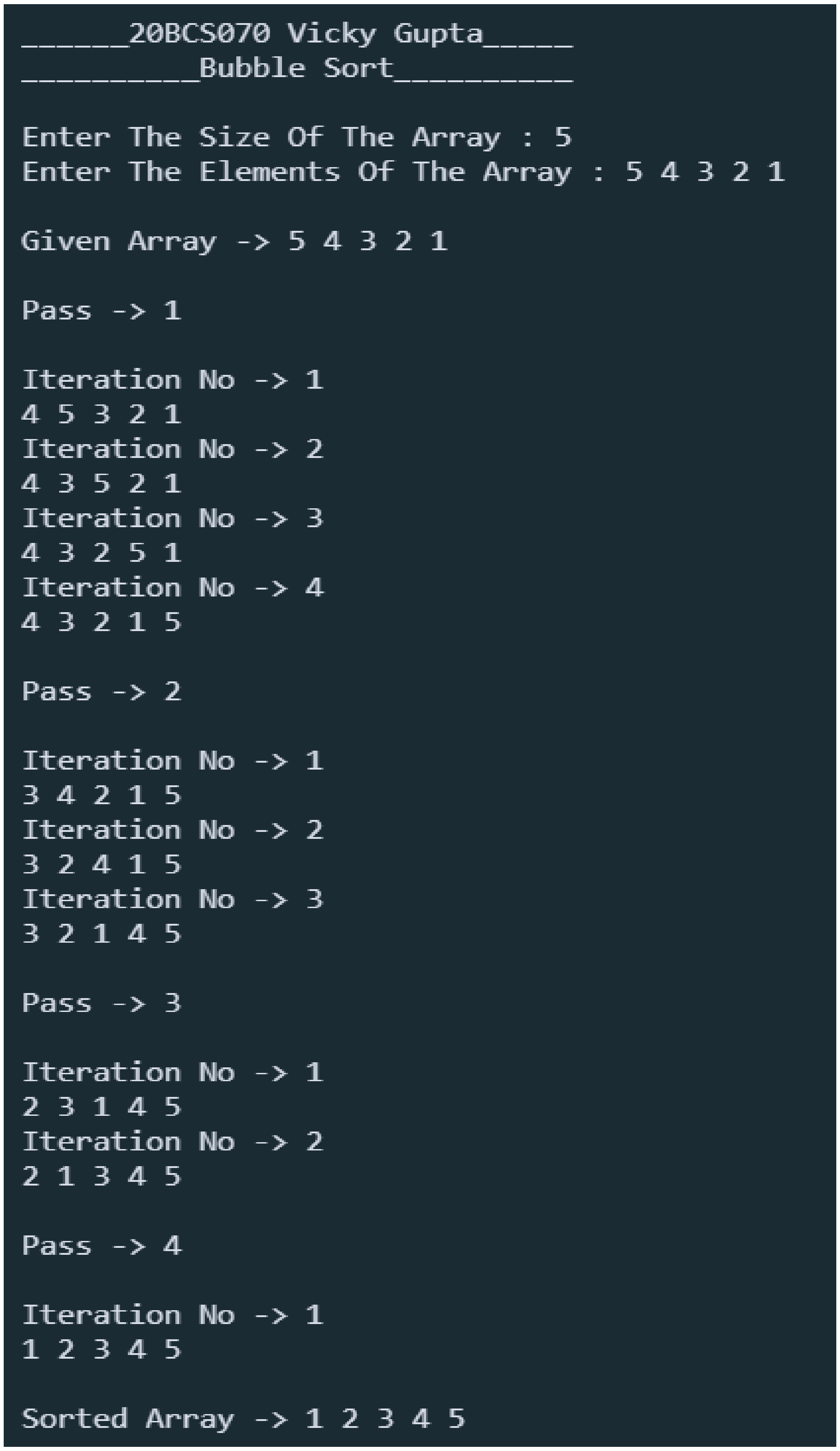
Bubble Sort

Code :-

```cpp
#include <iostream>
using namespace std;
#define size 1000

void Swap(int arr[], int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void PrintArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}
```

```cpp
void Bubble_Sort(int arr[], int n)
{
        cout << endl
         << "Given Array -> ";
    PrintArray(arr, n);
    for (int i = 1; i < n; i++)
    {
        cout << endl
             << "Pass -> " << i << endl<<endl;
        for (int j = 1; j < n + 1 - i; j++)
        {
            cout<< "Iteration No -> " << j << endl;
            if (arr[j - 1] > arr[j])
                Swap(arr, j, j - 1);
                PrintArray(arr, n);
        }
    }
}

int main()
{
    system("cls");
    cout<<"______20BCS070 Vicky Gupta_____"<<endl;
    cout<<"__________Bubble Sort__________"<<endl<<endl;
    int n, arr[size];

    cout << "Enter The Size Of The Array : ";
    cin >> n;

    cout << "Enter The Elements Of The Array : ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    Bubble_Sort(arr, n);

    cout << endl
         << "Sorted Array -> ";

    PrintArray(arr, n);
    cout<<endl;
    return 0;
}
```

Output :-



```
______20BCS070 Vicky Gupta_____
__________Bubble Sort__________

Enter The Size Of The Array : 5
Enter The Elements Of The Array : 5 4 3 2 1

Given Array -> 5 4 3 2 1

Pass -> 1

Iteration No -> 1
4 5 3 2 1
Iteration No -> 2
4 3 5 2 1
Iteration No -> 3
4 3 2 5 1
Iteration No -> 4
4 3 2 1 5

Pass -> 2

Iteration No -> 1
3 4 2 1 5
Iteration No -> 2
3 2 4 1 5
Iteration No -> 3
3 2 1 4 5

Pass -> 3

Iteration No -> 1
2 3 1 4 5
Iteration No -> 2
2 1 3 4 5

Pass -> 4

Iteration No -> 1
1 2 3 4 5

Sorted Array -> 1 2 3 4 5
```

Data Structure Lab

CEN-391

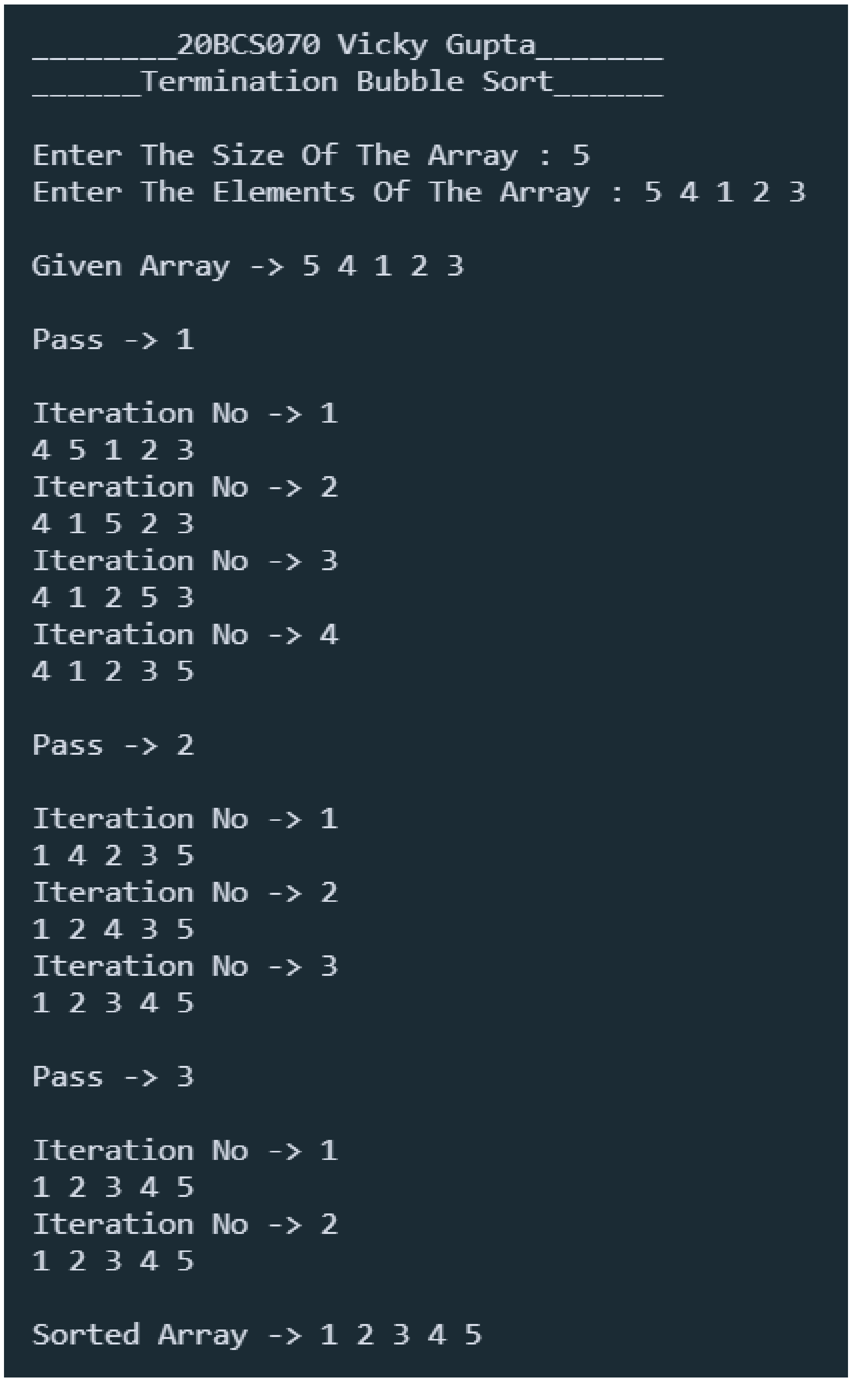
Early Termination Bubble Sort

Code :-

```
#include <iostream>
using namespace std;
#define size 1000

void Swap(int arr[], int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void PrintArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

void Bubble_Sort(int arr[], int n)
```

```cpp
{
    cout << endl
         << "Given Array -> ";
    PrintArray(arr, n);
    for (int i = 1; i < n; i++)
    {
        bool chk = true;
        cout << endl
             << "Pass -> " << i << endl
             << endl;
        for (int j = 1; j < n + 1 - i; j++)
        {
            cout << "Iteration No -> " << j << endl;
            if (arr[j - 1] > arr[j])
            {
                Swap(arr, j, j - 1);
                chk = false;
            }
            PrintArray(arr, n);
        }
        if (chk)
            break;
    }
}

int main()
{
    system("cls");
    cout<<"________20BCS070 Vicky Gupta_______"<<endl;
    cout << "______Termination Bubble Sort______" << endl
         << endl;
    int n, arr[size];

    cout << "Enter The Size Of The Array : ";
    cin >> n;

    cout << "Enter The Elements Of The Array : ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    Bubble_Sort(arr, n);

    cout << endl
         << "Sorted Array -> ";
    PrintArray(arr, n);
    cout << endl;
    return 0;
}
```

Output :-



```
________20BCS070 Vicky Gupta_______
______Termination Bubble Sort______

Enter The Size Of The Array : 5
Enter The Elements Of The Array : 5 4 1 2 3

Given Array -> 5 4 1 2 3

Pass -> 1

Iteration No -> 1
4 5 1 2 3
Iteration No -> 2
4 1 5 2 3
Iteration No -> 3
4 1 2 5 3
Iteration No -> 4
4 1 2 3 5

Pass -> 2

Iteration No -> 1
1 4 2 3 5
Iteration No -> 2
1 2 4 3 5
Iteration No -> 3
1 2 3 4 5

Pass -> 3

Iteration No -> 1
1 2 3 4 5
Iteration No -> 2
1 2 3 4 5

Sorted Array -> 1 2 3 4 5
```

Data Structure Lab

CEN-391

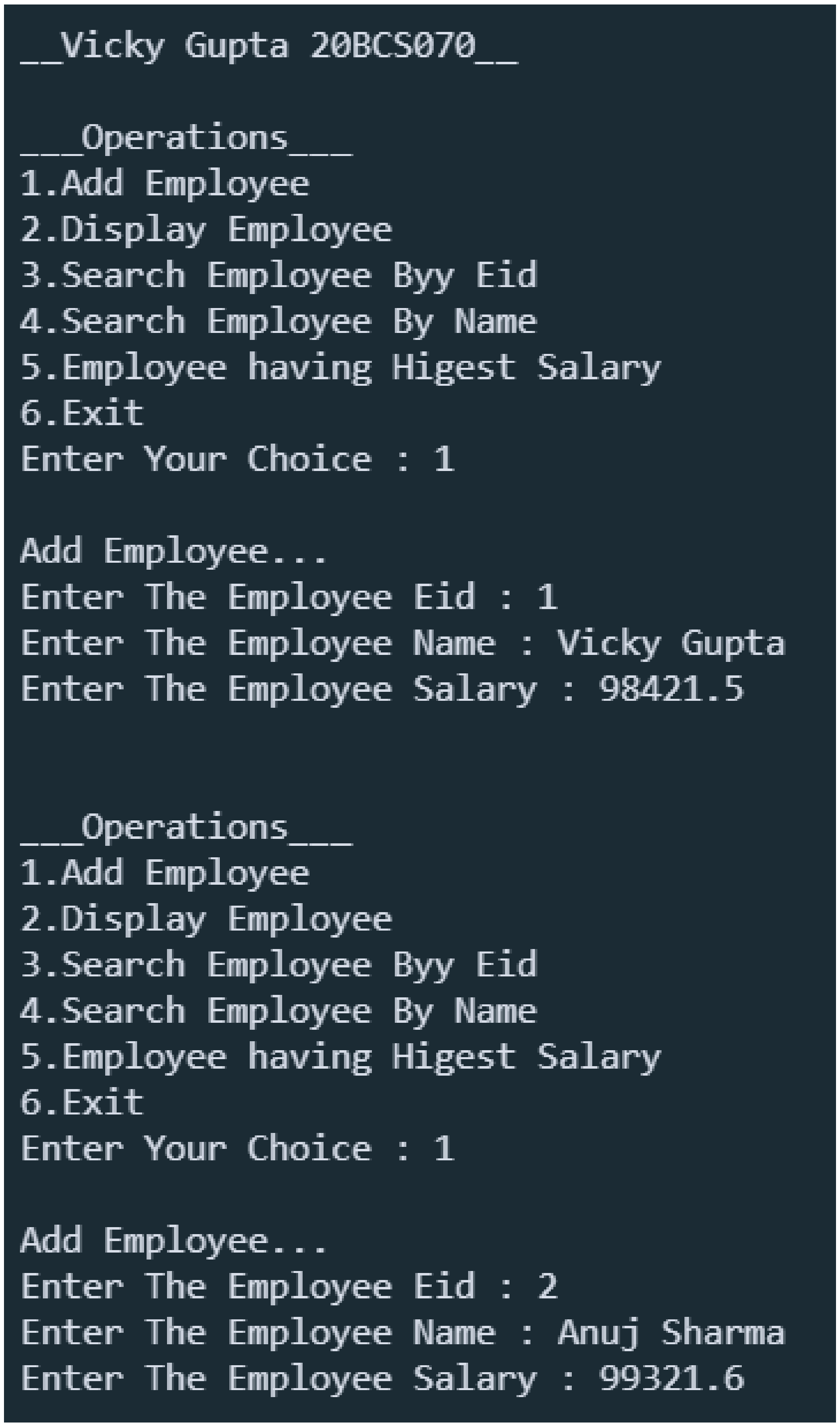
Program 3

Code :-

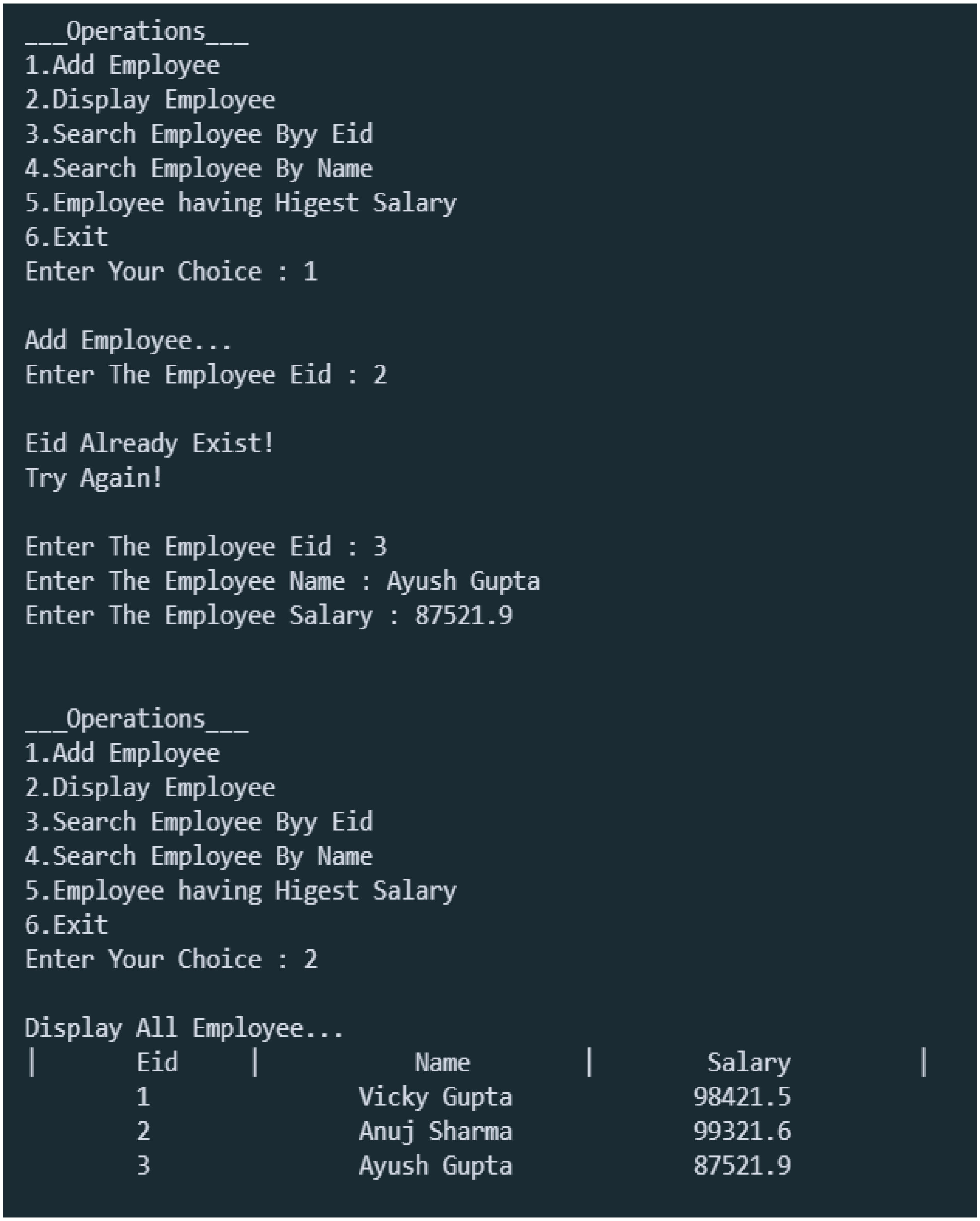
```
#include <iostream>
#include <string.h>
using namespace std;
#define Max_size 10
struct Employee
{
    int Eid;
    char Name[30];
    float Salary;
};

void Add_Employee(Employee Emp_Data[], int &size)
{
    cout << endl
         << "Add Employee..." << endl;
    if (size == Max_size)
    {
        cout << "Overflow" << endl;
        return;
    }
repeat:
    int Eid;
    cout << "Enter The Employee Eid : ";
    cin >> Eid;
    for (int i = 0; i < size; i++)
    {
        if (Eid == Emp_Data[i].Eid)
        {
            cout <<endl<< "Eid Already Exist!" << endl;
            cout << "Try Again!" << endl<<endl;
            goto repeat;
        }
    }
    Emp_Data[size].Eid=Eid;
    fflush(stdin);
    cout << "Enter The Employee Name : ";
    gets(Emp_Data[size].Name);
    cout << "Enter The Employee Salary : ";
    cin >> Emp_Data[size].Salary;
    size++;
}

void Display_Employee(Employee Emp_Data[], int &size)
{
    if (size == 0)
    {
        cout << "Empty!" << endl;
        return;
    }
    cout << endl
         << "Display All Employee..." << endl;
    cout << "|\tEid \t|"
         << "\t    Name     \t|"
         << "\t Salary \t|" << endl;
    for (int i = 0; i < size; i++)
    {
        cout << "\t" << Emp_Data[i].Eid << "\t";
        cout << "\t" << Emp_Data[i].Name << "\t";
        cout << "\t" << Emp_Data[i].Salary << "\t" << endl;
```

```cpp
    }
}

void Search_Employee_Eid(Employee Emp_Data[], int &size)
{
    cout << endl
         << "Search Employee By Eid..." << endl;
    if (size == 0)
    {
        cout << "Empty!" << endl;
        return;
    }
    int Eid;
    cout << "Enter The Employee Eid : ";
    cin >> Eid;
    int i;
    cout << endl;
    for (i = 0; i < size; i++)
    {
        if (Emp_Data[i].Eid == Eid)
        {
            cout << "Employee Found!\n\nDetails..." << endl;
            cout << "Eid : " << Emp_Data[i].Eid << "\t  ";
            cout << "Name : " << Emp_Data[i].Name << "\t  ";
            cout << "Salary : " << Emp_Data[i].Salary << endl;
            break;
        }
    }
    if (i == size)
    {
        cout << "Employee Not Found!" << endl;
    }
}

void Search_Employee_Name(Employee Emp_Data[], int &size)
{
    cout << endl
         << "Search Employee By Name..." << endl;
    if (size == 0)
    {
        cout << "Empty!" << endl;
        return;
    }
    char Name[30];
    cout << "Enter The Name Of Your Employee : ";
    fflush(stdin);
    gets(Name);
    int i;
    cout << endl;
    for (i = 0; i < size; i++)
    {
        int j;
        if (!strcmp(Name, Emp_Data[i].Name))
        {
            cout << "Employee Found!\n\nDetails..." << endl;
            cout << "Eid : " << Emp_Data[i].Eid << "\t  ";
            cout << "Name : " << Emp_Data[i].Name << "\t  ";
            cout << "Salary : " << Emp_Data[i].Salary << endl;
            break;
        }
    }
    if (i == size)
    {
        cout << "Employee Not Found!" << endl;
    }
}

void Highest_Salary(Employee Emp_Data[], int &size)
{
    cout << endl
         << "Highest Salary Of Employee" << endl;
    if (size == 0)
    {
        cout << "Empty!" << endl;
        return;
    }
    float Max_Salary = 0;
    for (int i = 0; i < size; i++)
    {
        if (Max_Salary < Emp_Data[i].Salary)
        {
            Max_Salary = Emp_Data[i].Salary;
        }
    }
    for (int i = 0; i < size; i++)
    {
        if (Max_Salary == Emp_Data[i].Salary)
        {
            cout << "Eid : " << Emp_Data[i].Eid << "\t  ";
            cout << "Name : " << Emp_Data[i].Name << "\t ";
            cout << "Salary : " << Emp_Data[i].Salary << endl;
        }
    }
}

void Menu()
{
    cout << endl
         << endl
         << "___Operations___" << endl;
    cout << "1.Add Employee" << endl;
    cout << "2.Display Employee" << endl;
    cout << "3.Search Employee Byy Eid" << endl;
    cout << "4.Search Employee By Name" << endl;
    cout << "5.Employee having Higest Salary" << endl;
    cout << "6.Exit" << endl;
    cout << "Enter Your Choice : ";
}

bool Options(Employee Emp_Data[], int &size)
{
    int opt;
    cin >> opt;
    switch (opt)
    {
    case 1:
        Add_Employee(Emp_Data, size);
        break;
    case 2:
        Display_Employee(Emp_Data, size);
        break;
    case 3:
        Search_Employee_Eid(Emp_Data, size);
        break;
    case 4:
        Search_Employee_Name(Emp_Data, size);
        break;
    case 5:
        Highest_Salary(Emp_Data, size);
        break;
    case 6:
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!" << endl;
    }
    return 1;
}

int main()
{
    system("cls");
    cout << "__Vicky Gupta 20BCS070__";
    struct Employee Emp_Data[Max_size];
    int size = 0;
    while (true)
    {
        Menu();
        if (!Options(Emp_Data, size))
            break;
    }
    cout<<"Exiting..."<<endl;
    return 0;
}
```

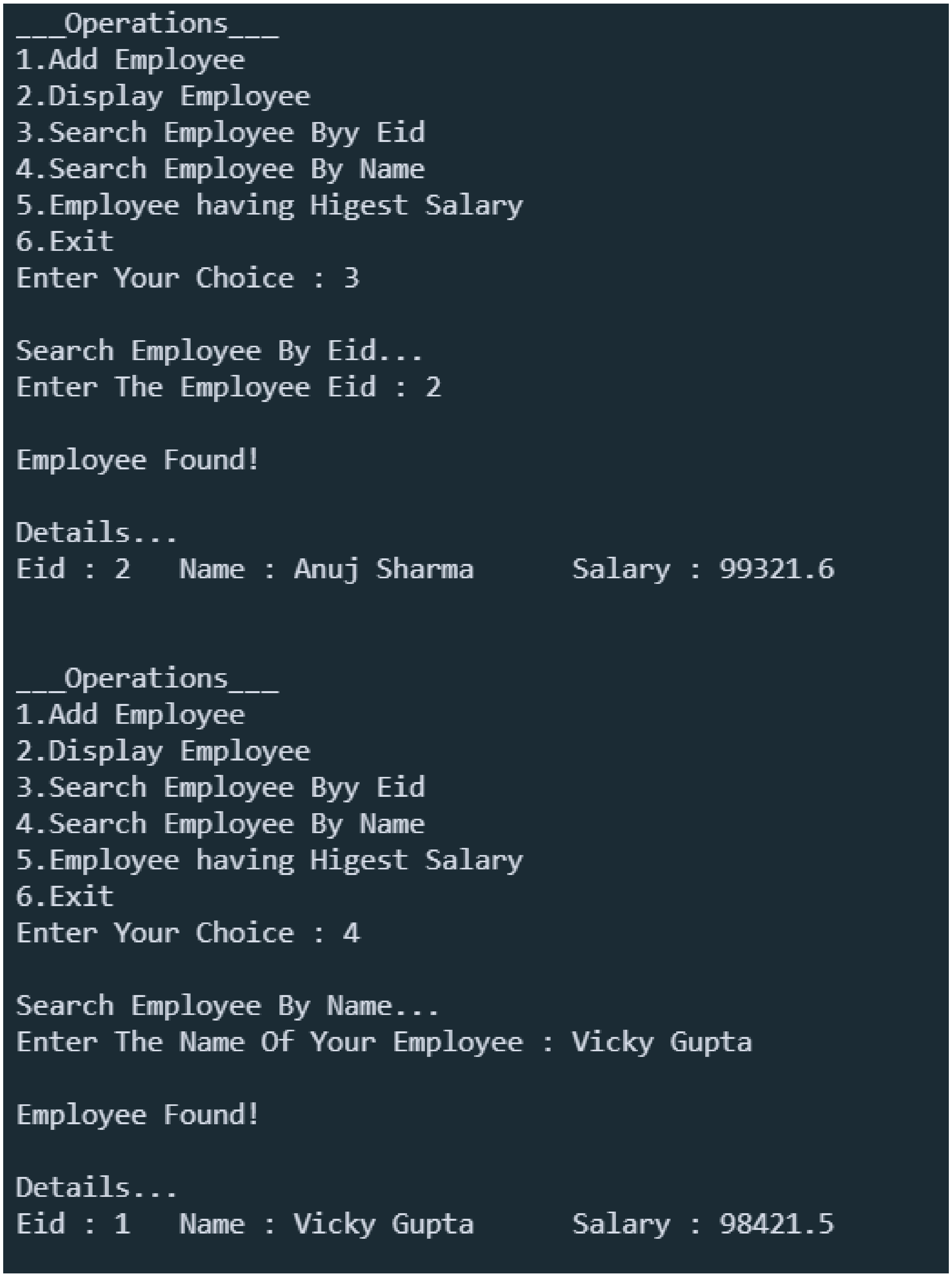
Output :-



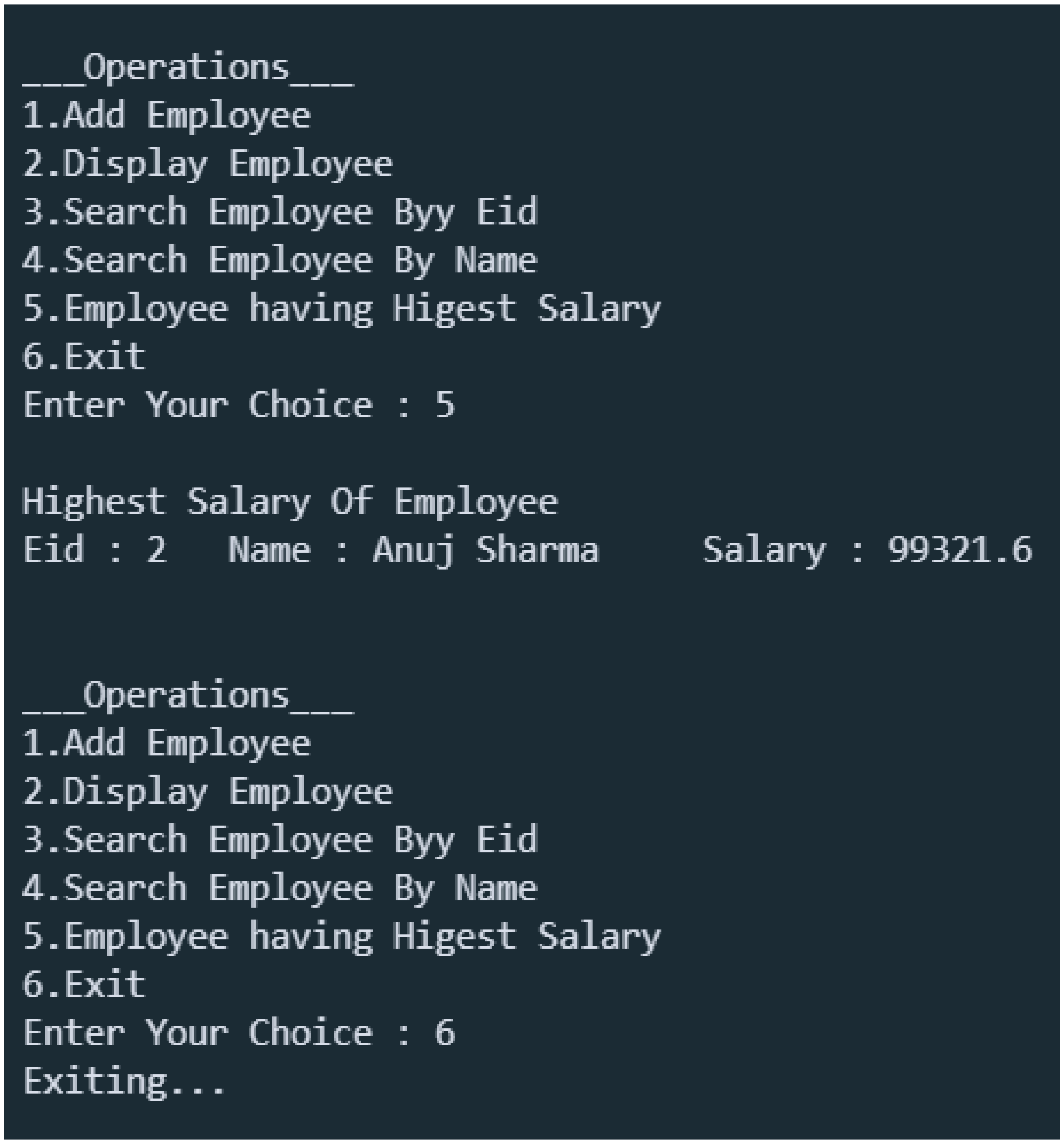
```
__Vicky Gupta 20BCS070__

___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 1
Enter The Employee Name : Vicky Gupta
Enter The Employee Salary : 98421.5


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 2
Enter The Employee Name : Anuj Sharma
Enter The Employee Salary : 99321.6
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 2

Eid Already Exist!
Try Again!

Enter The Employee Eid : 3
Enter The Employee Name : Ayush Gupta
Enter The Employee Salary : 87521.9


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 2

Display All Employee...
|       Eid     |           Name        |        Salary          |
        1             Vicky Gupta             98421.5
        2             Anuj Sharma             99321.6
        3             Ayush Gupta             87521.9
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 3

Search Employee By Eid...
Enter The Employee Eid : 2

Employee Found!

Details...
Eid : 2   Name : Anuj Sharma      Salary : 99321.6


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 4

Search Employee By Name...
Enter The Name Of Your Employee : Vicky Gupta

Employee Found!

Details...
Eid : 1   Name : Vicky Gupta      Salary : 98421.5
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 5

Highest Salary Of Employee
Eid : 2   Name : Anuj Sharma     Salary : 99321.6


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Exit
Enter Your Choice : 6
Exiting...
```

Data Structure Lab

CEN-391

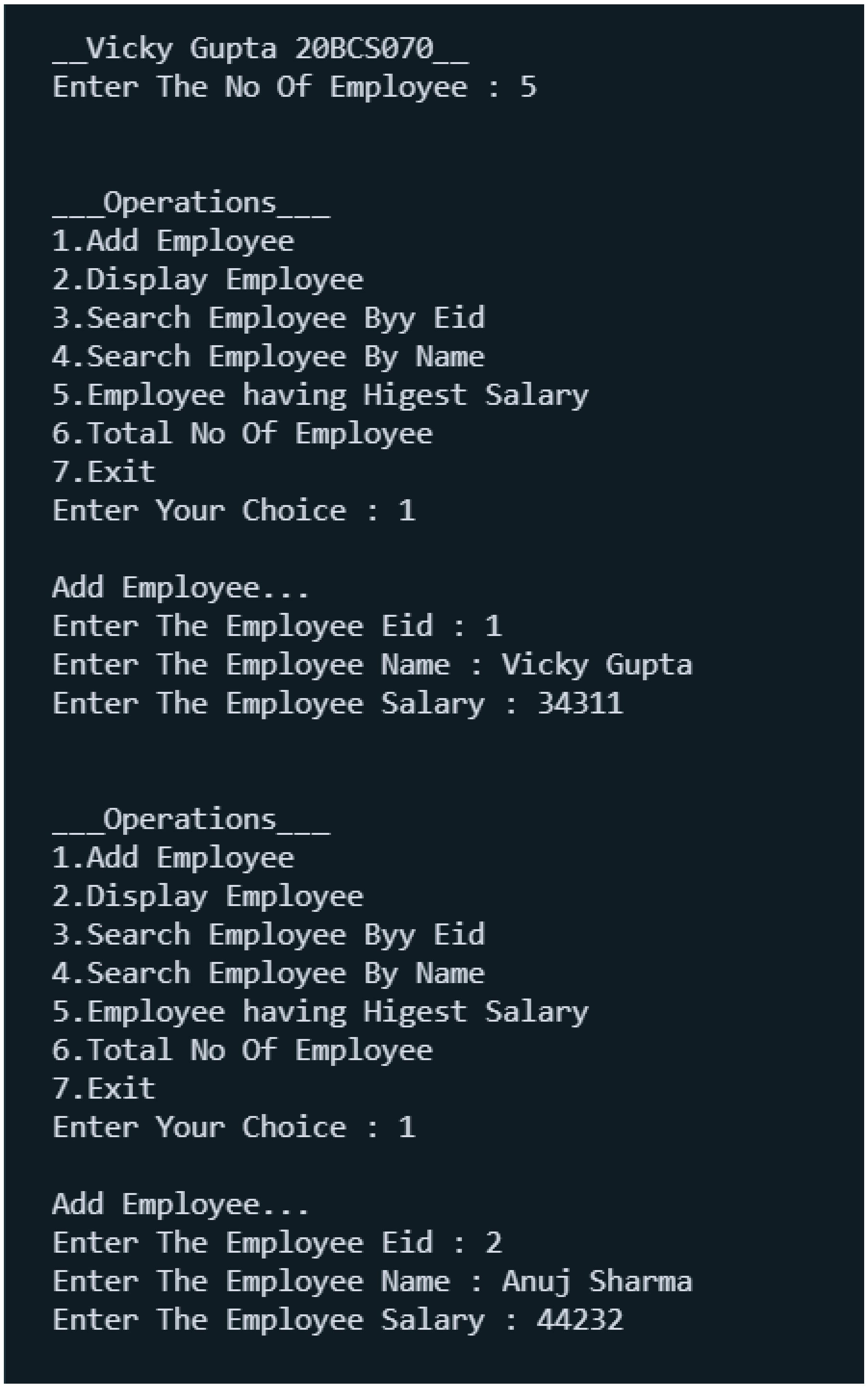
Program 4

Code :-

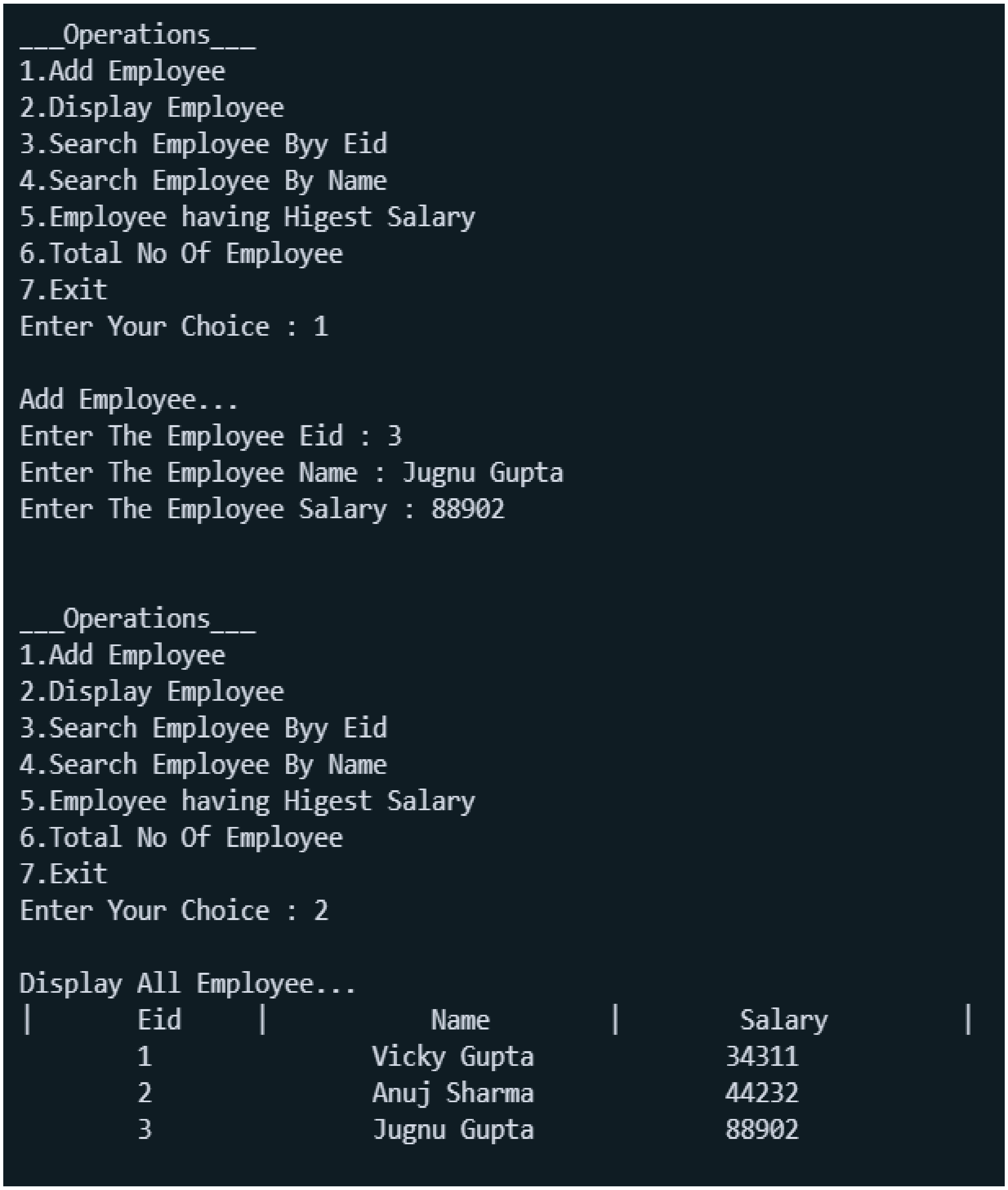
```
#include <iostream>
#include <string.h>
using namespace std;
int Max_Size = 0;
struct Employee
{
    int Eid;
    char Name[30];
    float Salary;
};

void Add_Employee(Employee *Emp_Data, int &size)
{
    cout << endl
         << "Add Employee..." << endl;
    if (size == Max_Size)
    {
        cout << "Overflow" << endl;
```

```
            return;
        }
        int Eid;
        bool check = false;
        do
        {
            cout << "Enter The Employee Eid : ";
            cin >> Eid;
            for (int i = 0; i < size; i++)
            {
                if (Eid == (Emp_Data + i)->Eid)
                {
                    cout << endl
                         << "Eid Already Exist!" << endl;
                    cout << "Try Again!" << endl
                         << endl;
                    check = true;
                }
            }
        } while (check);

        (Emp_Data + size)->Eid = Eid;
        fflush(stdin);
        cout << "Enter The Employee Name : ";
        gets((Emp_Data + size)->Name);
        cout << "Enter The Employee Salary : ";
        cin >> (Emp_Data + size)->Salary;
        size++;
    }

    void Display_Employee(Employee *Emp_Data, int &size)
    {
        if (size == 0)
        {
            cout << endl
                  << "Empty!" << endl;
            return;
        }
        cout << endl
             << "Display All Employee..." << endl;
        cout << "|\tEid \t|"
             << "\t    Name     \t|"
             << "\t Salary \t|" << endl;
        for (int i = 0; i < size; i++)
        {
            cout << "\t" << (Emp_Data + i)->Eid << "\t";
            cout << "\t" << (Emp_Data + i)->Name << "\t";
            cout << "\t" << (Emp_Data + i)->Salary << "\t" << endl;
        }
    }

    void Search_Employee_Eid(Employee *Emp_Data, int &size)
    {
        cout << endl
             << "Search Employee By Eid..." << endl;
        if (size == 0)
        {
            cout << "Empty!" << endl;
            return;
        }
        int Eid;
        cout << "Enter The Employee Eid : ";
        cin >> Eid;
        int i;
        cout << endl;
        for (i = 0; i < size; i++)
        {
            if ((Emp_Data + i)->Eid == Eid)
            {
                cout << "Employee Found!\n\nDetails..." << endl;
                cout << "Eid : " << (Emp_Data + i)->Eid << "\t  ";
                cout << "Name : " << (Emp_Data + i)->Name << "\t  ";
                cout << "Salary : " << (Emp_Data + i)->Salary << endl;
                break;
            }
        }
        if (i == size)
        {
            cout << "Employee Not Found!" << endl;
        }
    }

    void Search_Employee_Name(Employee *Emp_Data, int &size)
    {
        cout << endl
             << "Search Employee By Name..." << endl;
        if (size == 0)
        {
            cout << "Empty!" << endl;
            return;
        }
        char Name[30];
        cout << "Enter The Name Of Your Employee : ";
        fflush(stdin);
        gets(Name);
        int i;
        cout << endl;
        for (i = 0; i < size; i++)
        {
            int j;
            if (!strcmp(Name, (Emp_Data + i)->Name))
            {
                cout << "Employee Found!\n\nDetails..." << endl;
                cout << "Eid : " << (Emp_Data + i)->Eid << "\t  ";
                cout << "Name : " << (Emp_Data + i)->Name << "\t  ";
                cout << "Salary : " << (Emp_Data + i)->Salary << endl;
                break;
            }
        }
        if (i == size)
        {
            cout << "Employee Not Found!" << endl;
        }
    }

    void Highest_Salary(Employee *Emp_Data, int &size)
    {
        cout << endl
             << "Highest Salary Of Employee" << endl;
        if (size == 0)
        {
            cout << "Empty!" << endl;
            return;
        }
        float Max_Salary = 0;
        for (int i = 0; i < size; i++)
        {
            if (Max_Salary < (Emp_Data + i)->Salary)
            {
                Max_Salary = (Emp_Data + i)->Salary;
            }
        }
        for (int i = 0; i < size; i++)
        {
            if (Max_Salary == (Emp_Data + i)->Salary)
            {
                cout << "Eid : " << (Emp_Data + i)->Eid << "\t  ";
                cout << "Name : " << (Emp_Data + i)->Name << "\t ";
                cout << "Salary : " << (Emp_Data + i)->Salary << endl;
            }
        }
    }

    void Total_Employee(int &size)
    {
        cout << endl
             << "No Of Employee..." << endl;
        cout << endl
             << "Total No Of Employee : ";
        cout << size << endl;
    }

    void Menu()
    {
        cout << endl
             << endl
             << "___Operations___" << endl;
        cout << "1.Add Employee" << endl;
        cout << "2.Display Employee" << endl;
        cout << "3.Search Employee Byy Eid" << endl;
        cout << "4.Search Employee By Name" << endl;
        cout << "5.Employee having Higest Salary" << endl;
        cout << "6.Total No Of Employee" << endl;
        cout << "7.Exit" << endl;
        cout << "Enter Your Choice : ";
    }

    bool Options(Employee *Emp_Data, int &size)
    {
        int opt;
        cin >> opt;
        switch (opt)
        {
        case 1:
            Add_Employee(Emp_Data, size);
            break;
        case 2:
            Display_Employee(Emp_Data, size);
            break;
        case 3:
            Search_Employee_Eid(Emp_Data, size);
            break;
        case 4:
            Search_Employee_Name(Emp_Data, size);
            break;
        case 5:
            Highest_Salary(Emp_Data, size);
            break;
        case 6:
            Total_Employee(size);
            break;
        case 7:
            return 0;
        default:
            cout << "Invalid Input!\nTry Again!" << endl;
        }
        return 1;
    }

    int main()
    {
        system("cls");
        cout << "__Vicky Gupta 20BCS070__" << endl;

        cout << "Enter The No Of Employee : ";
        cin >> Max_Size;

        struct Employee *Emp_Data = (Employee *)malloc(Max_Size *
    sizeof(Employee));

        int size = 0;

        while (true)
        {
            Menu();
            if (!Options(Emp_Data, size))
                break;
        }
        cout << endl
             << "Exiting..." << endl;
        return 0;
    }
```

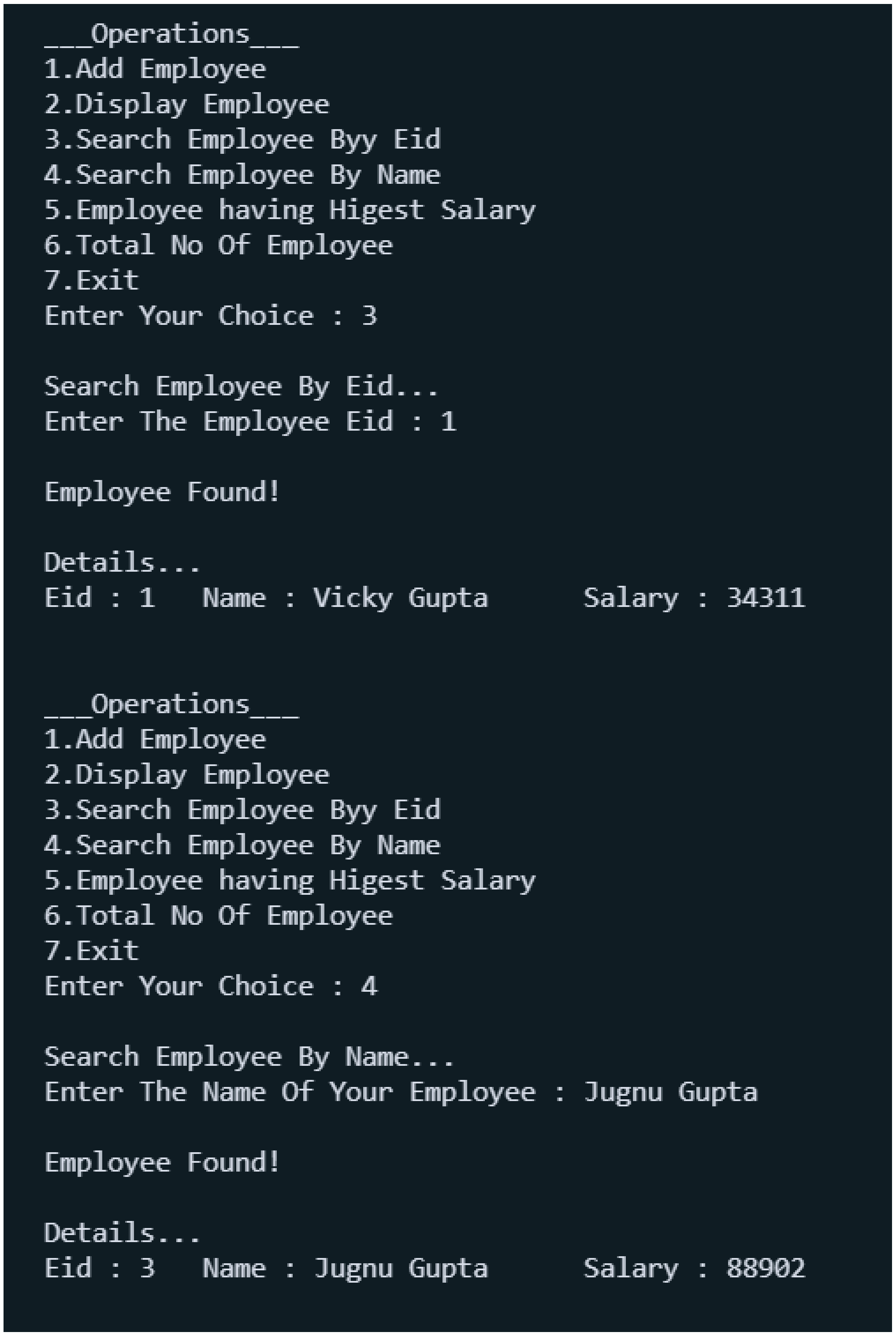
Output :-



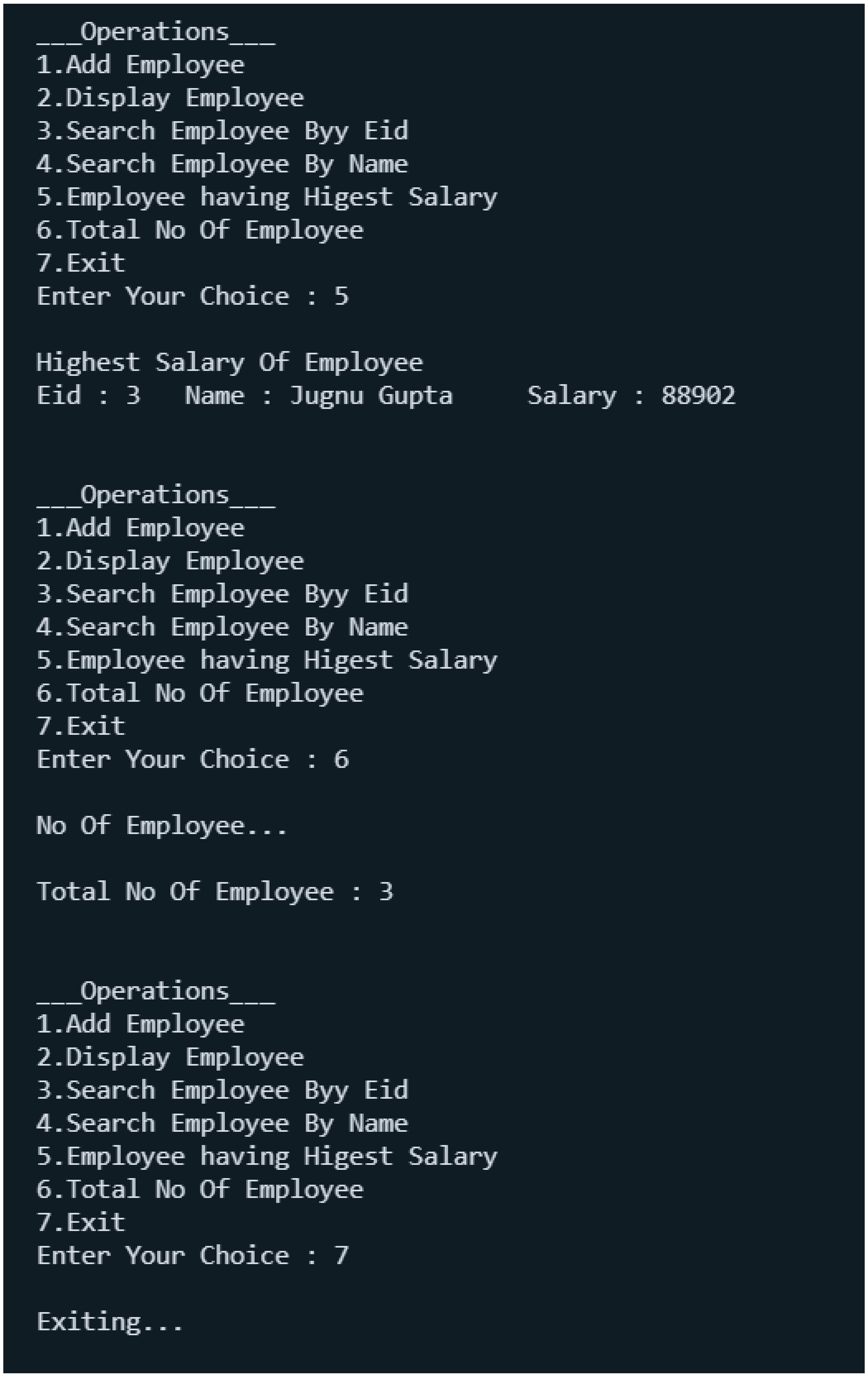
```
__Vicky Gupta 20BCS070__
Enter The No Of Employee : 5


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 1
Enter The Employee Name : Vicky Gupta
Enter The Employee Salary : 34311


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 2
Enter The Employee Name : Anuj Sharma
Enter The Employee Salary : 44232
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1

Add Employee...
Enter The Employee Eid : 3
Enter The Employee Name : Jugnu Gupta
Enter The Employee Salary : 88902


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 2

Display All Employee...
|       Eid     |           Name         |         Salary         |
        1              Vicky Gupta              34311
        2              Anuj Sharma              44232
        3              Jugnu Gupta              88902
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 3

Search Employee By Eid...
Enter The Employee Eid : 1

Employee Found!

Details...
Eid : 1   Name : Vicky Gupta      Salary : 34311


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 4

Search Employee By Name...
Enter The Name Of Your Employee : Jugnu Gupta

Employee Found!

Details...
Eid : 3   Name : Jugnu Gupta      Salary : 88902
```


```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 5

Highest Salary Of Employee
Eid : 3   Name : Jugnu Gupta     Salary : 88902


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 6

No Of Employee...

Total No Of Employee : 3


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 7

Exiting...
```

Data Structure Lab

CEN-391

Program 5

Code :-

```
#include <iostream>
#include <string.h>
using namespace std;

struct Employee
{
    int Eid;
    char Name[30];
    float Salary;
    struct Employee *next;
};

void Add_Employee(Employee *&Emp_Data, int &size)
{
    cout << "Add Employee..." << endl;
    struct Employee *newEmployee = (Employee
*)malloc(sizeof(Employee));
```

```
        int Eid;
        bool check = false;
        do
        {
            cout << "Enter The Employee Eid : ";
            cin >> Eid;
            Employee*temp=Emp_Data;
            while(temp!=nullptr)
            {
                if (Eid == temp->Eid)
                {
                    cout << endl
                         << "Eid Already Exist!" << endl;
                    cout << "Try Again!" << endl
                         << endl;
                    check = true;
                }
                temp=temp->next;
            }
        } while (check);

        newEmployee->Eid = Eid;
        fflush(stdin);
        cout << "Enter The Employee Name : ";
        gets(newEmployee->Name);
        cout << "Enter The Employee Salary : ";
        cin >> newEmployee->Salary;

        newEmployee->next = Emp_Data;
        Emp_Data = newEmployee;
        size++;
    }

    void Display_Employee(Employee *Emp_Data, int &size)
    {
        if (size == 0)
        {
            cout << endl
                 << "Empty!" << endl;
            return;
        }
        cout <<  "Display All Employee..." << endl;
        cout << "|Eid\t\t|"
             << "Name\t\t|"
             << "Salary\t\t|" << endl;

        Employee *temp = Emp_Data;

        while (temp != nullptr)
        {
            cout << "\t" << temp->Eid << "\t";
            cout << temp->Name << "\t";
            cout << temp->Salary << "\t" << endl;
            temp = temp->next;
        }
    }

    void Search_Employee_Eid(Employee *Emp_Data, int &size)
    {
        cout << "Search Employee By Eid..." << endl;
        if (size == 0)
        {
            cout << "Empty!" << endl;
            return;
        }
        int Eid;
        cout << "Enter The Employee Eid : ";
        cin >> Eid;
        cout << endl;
        Employee *temp = Emp_Data;
        while (temp != nullptr)
        {
            if (temp->Eid == Eid)
            {
                cout << "Employee Found!\n\nDetails..." << endl;
                cout << "Eid : " << temp->Eid << "\t  ";
                cout << "Name : " << temp->Name << "\t  ";
                cout << "Salary : " << temp->Salary << endl;
                break;
            }
            temp = temp->next;
        }
        if (temp == nullptr)
        {
```

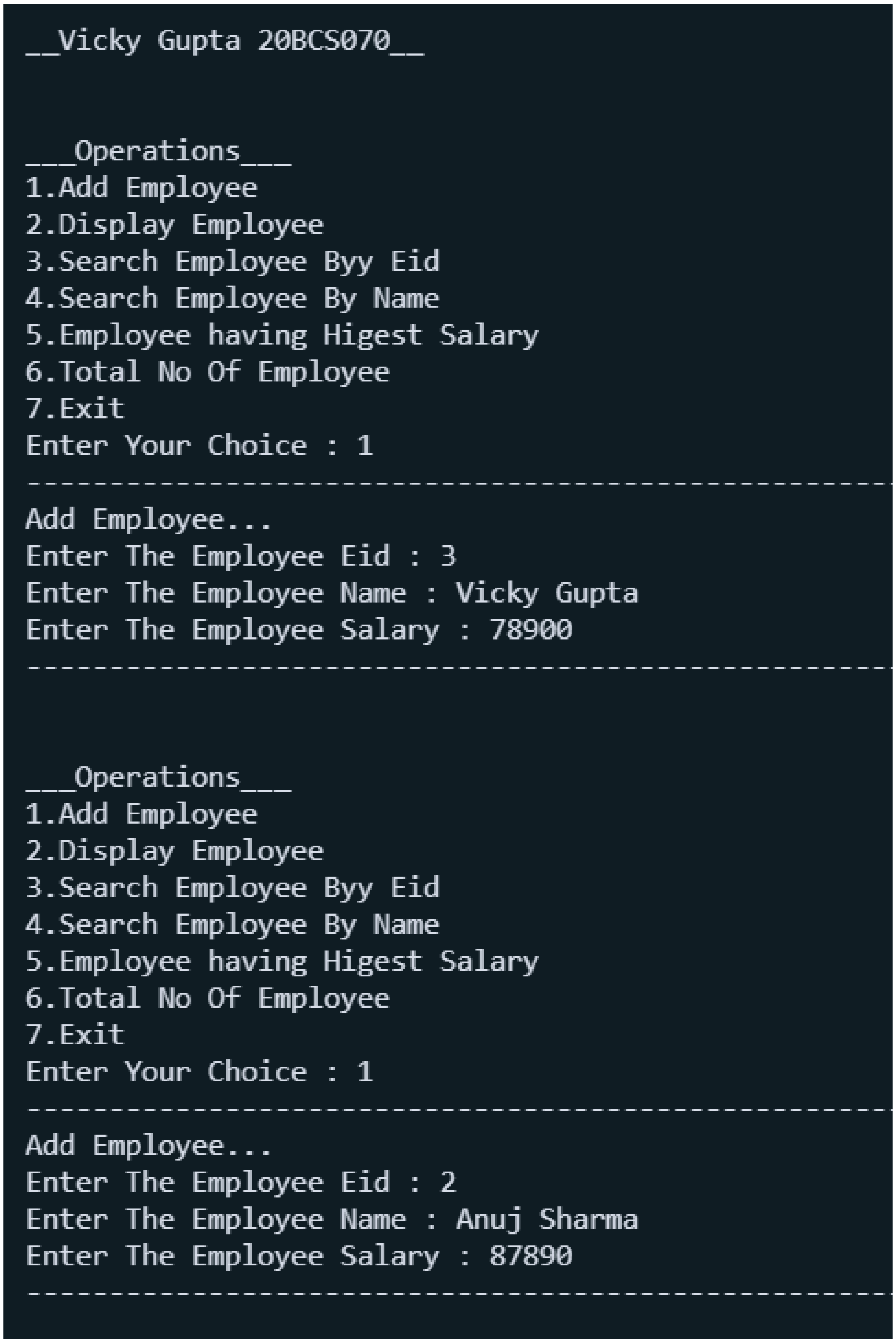
```
        cout << "Employee Not Found!" << endl;
    }
}

void Search_Employee_Name(Employee *Emp_Data, int &size)
{
    cout << "Search Employee By Name..." << endl;
    if (size == 0)
    {
        cout << "Empty!" << endl;
        return;
    }
    char Name[30];
    cout << "Enter The Name Of Your Employee : ";
    fflush(stdin);
    gets(Name);
    cout << endl;
    Employee *temp = Emp_Data;
    while (temp != nullptr)
    {
        if (!strcmp(Name, temp->Name))
        {
            cout << "Employee Found!\n\nDetails..." << endl;
            cout << "Eid : " << temp->Eid << "\t  ";
            cout << "Name : " << temp->Name << "\t  ";
            cout << "Salary : " << temp->Salary << endl;
            break;
        }
        temp = temp->next;
    }
    if (temp == nullptr)
    {
        cout << "Employee Not Found!" << endl;
    }
}

void Highest_Salary(Employee *Emp_Data, int &size)
{
    cout << "Highest Salary Of Employee" << endl;
    if (size == 0)
    {
        cout << "Empty!" << endl;
```

```
            return;
        }
        Employee *temp = Emp_Data->next, *MaxEmployee = Emp_Data;

        while (temp != nullptr)
        {
            if (MaxEmployee->Salary < temp->Salary)
            {
                MaxEmployee = temp;
            }
            temp=temp->next;
        }
        temp = Emp_Data;
        while (temp != nullptr)
        {
            if (MaxEmployee->Salary == temp->Salary)
            {
                cout << "Eid : " << temp->Eid << "\t  ";
                cout << "Name : " << temp->Name << "\t ";
                cout << "Salary : " << temp->Salary << endl;
            }
            temp=temp->next;
        }
    }

    void Total_Employee(int &size)
    {
        cout << endl
             << "No Of Employee..." << endl;
        cout << endl
             << "Total No Of Employee : ";
        cout << size << endl;
    }

    void AnsBar()
    {
        cout<<"------------------------------------------------------
    -------------------------\n";
    }

    void Menu()
    {
```

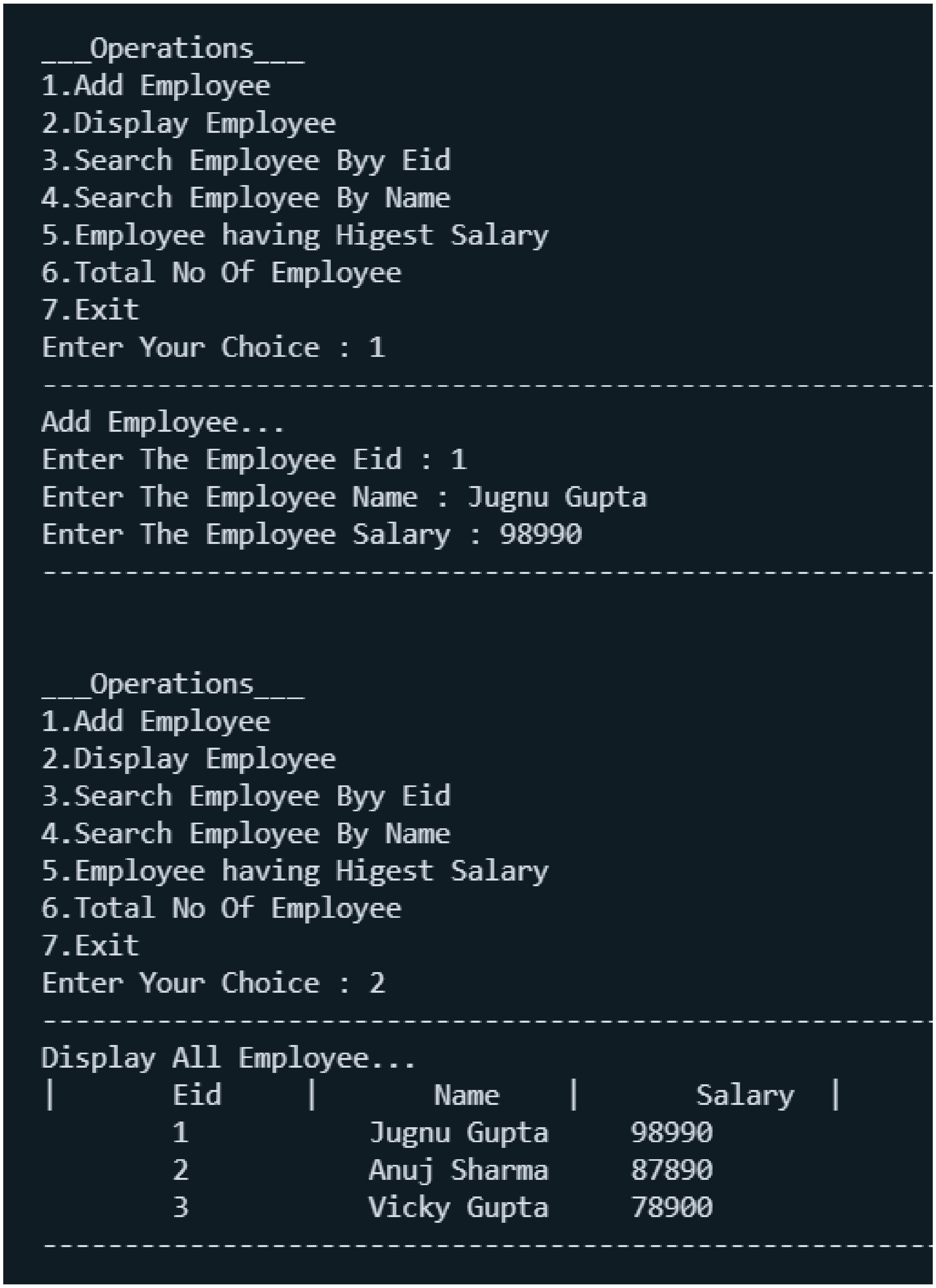
```
    cout << endl
         << endl
         << "___Operations___" << endl;
    cout << "1.Add Employee" << endl;
    cout << "2.Display Employee" << endl;
    cout << "3.Search Employee Byy Eid" << endl;
    cout << "4.Search Employee By Name" << endl;
    cout << "5.Employee having Higest Salary" << endl;
    cout << "6.Total No Of Employee" << endl;
    cout << "7.Exit" << endl;
    cout << "Enter Your Choice : ";
}

bool Options(Employee *&Emp_Data, int &size)
{
    int opt;
    cin >> opt;
    switch (opt)
    {
    case 1:AnsBar();
        Add_Employee(Emp_Data, size);
        break;
    case 2:AnsBar();
        Display_Employee(Emp_Data, size);
        break;
    case 3:AnsBar();
        Search_Employee_Eid(Emp_Data, size);
        break;
    case 4:AnsBar();
        Search_Employee_Name(Emp_Data, size);
        break;
    case 5:AnsBar();
        Highest_Salary(Emp_Data, size);
        break;
    case 6:AnsBar();
        Total_Employee(size);
        break;
    case 7:AnsBar();
    cout<<"Exit Operation Is Selected"<<endl;
    AnsBar();
        return 0;
    default:
```

```cpp
        cout << "Invalid Input!\nTry Again!" << endl;
    }
    AnsBar();
    return 1;
}

int main()
{
    system("cls");
    cout << "__Vicky Gupta 20BCS070__" << endl;

    struct Employee *Emp_Data = nullptr;
    int size = 0;

    while (true)
    {
        Menu();
        if (!Options(Emp_Data, size))
            break;
    }
    cout << endl
         << "Exiting..." << endl;
    return 0;
}
```

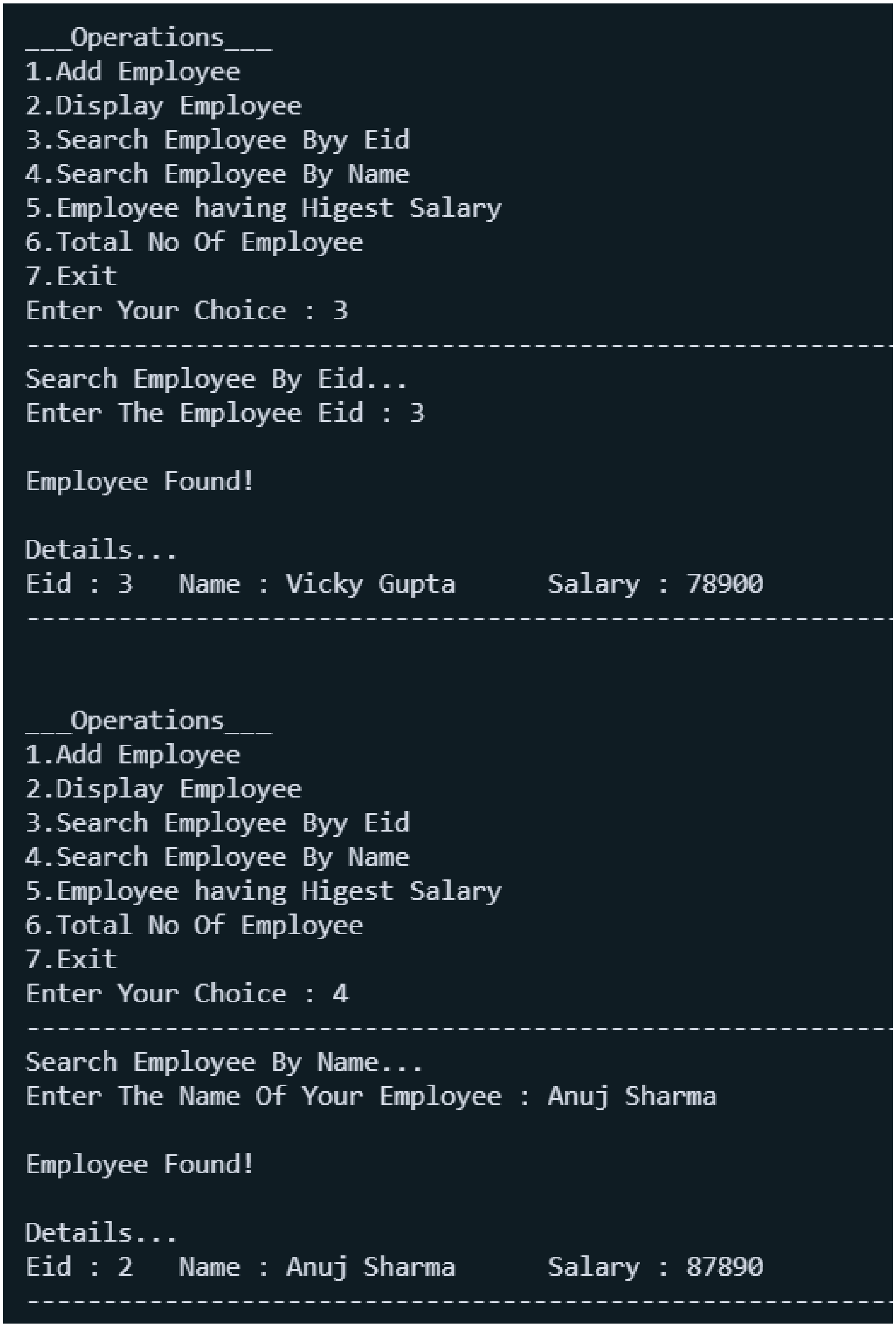
Output :-



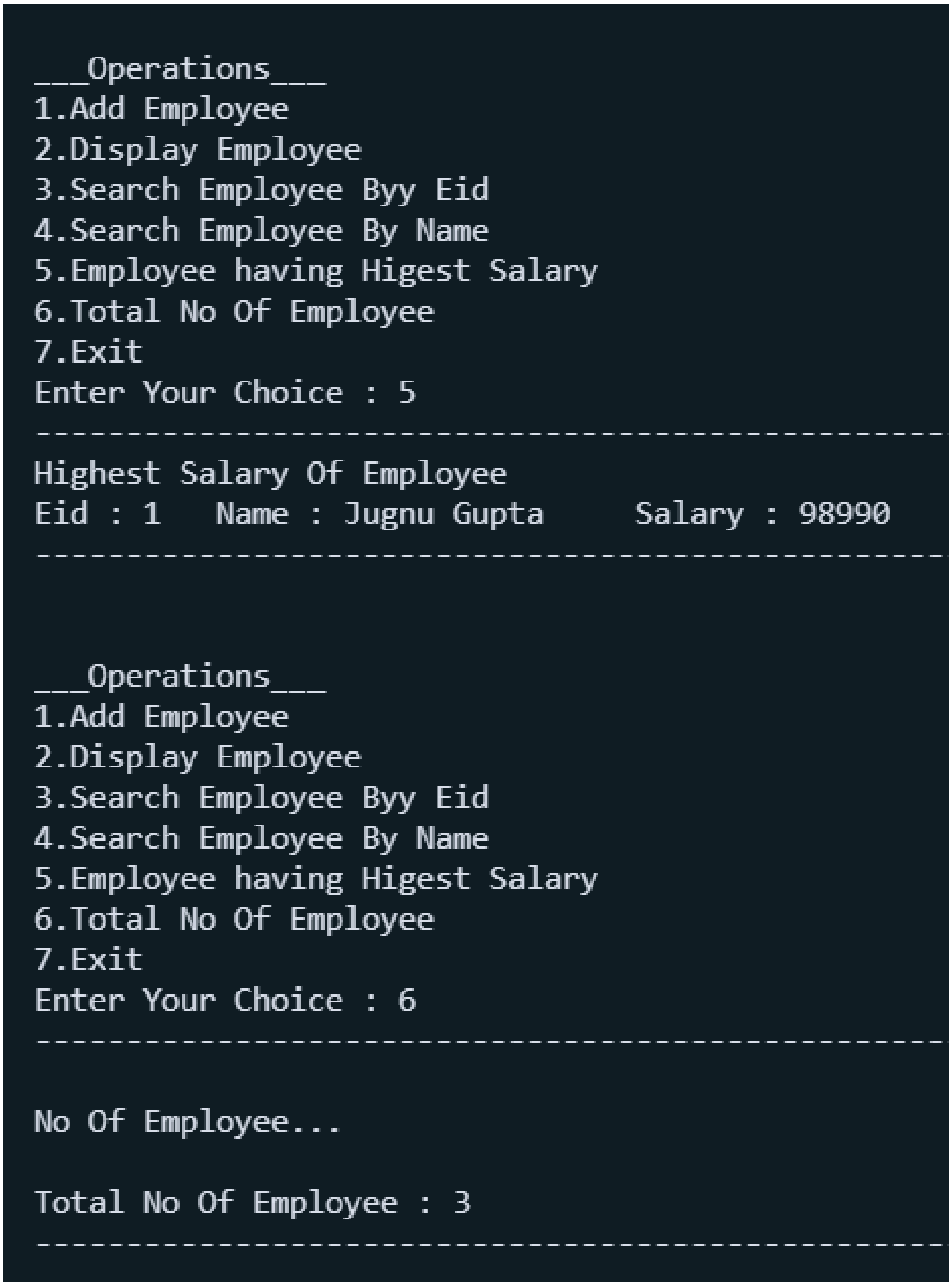
```
__Vicky Gupta 20BCS070__


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1
-----------------------------------------------------
Add Employee...
Enter The Employee Eid : 3
Enter The Employee Name : Vicky Gupta
Enter The Employee Salary : 78900
-----------------------------------------------------



___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1
-----------------------------------------------------
Add Employee...
Enter The Employee Eid : 2
Enter The Employee Name : Anuj Sharma
Enter The Employee Salary : 87890
-----------------------------------------------------
```



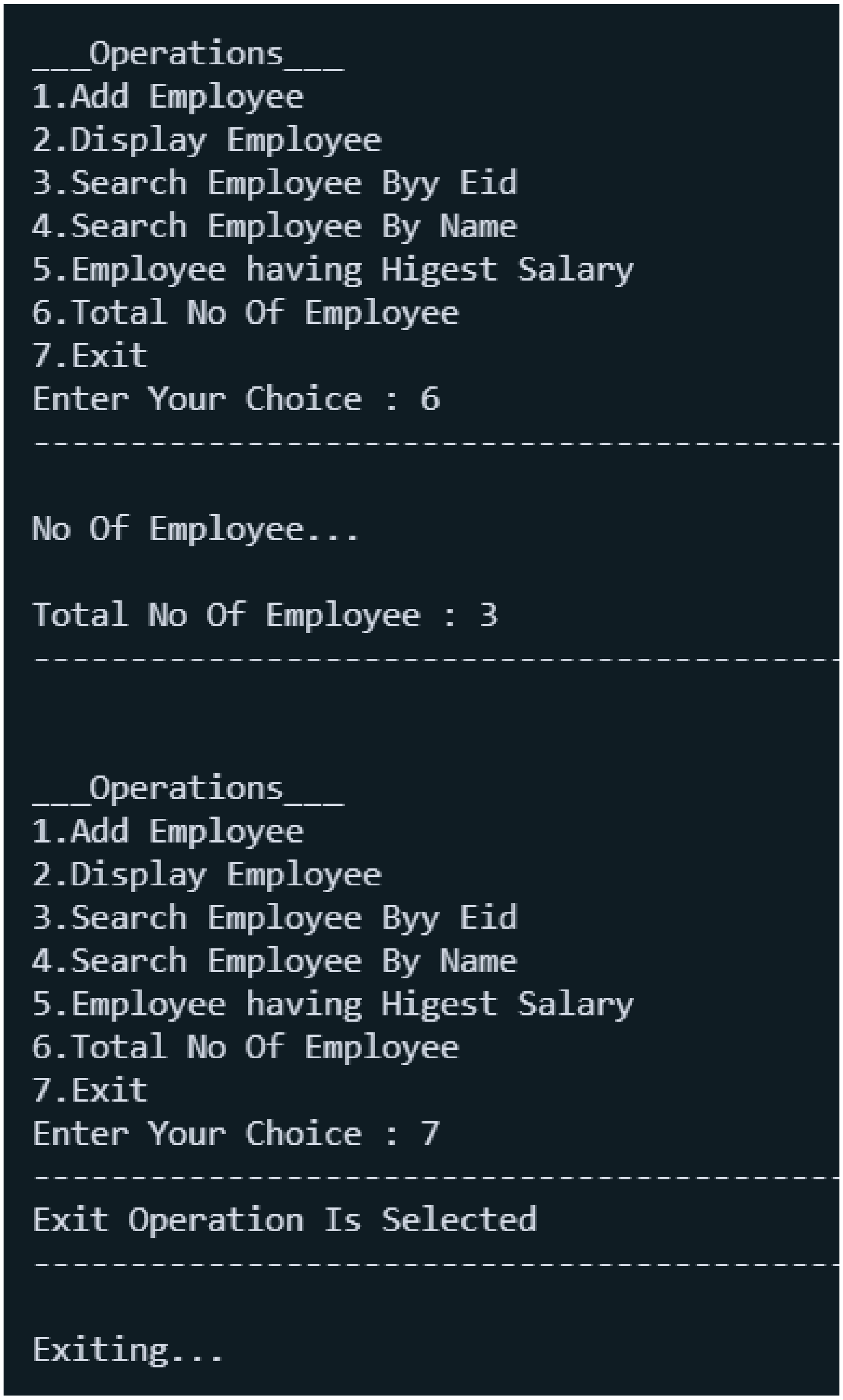
```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 1
-------------------------------------------------------
Add Employee...
Enter The Employee Eid : 1
Enter The Employee Name : Jugnu Gupta
Enter The Employee Salary : 98990
-------------------------------------------------------


___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 2
-------------------------------------------------------
Display All Employee...
|       Eid     |       Name    |       Salary  |
        1           Jugnu Gupta     98990
        2           Anuj Sharma     87890
        3           Vicky Gupta     78900
-------------------------------------------------------
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 3
---------------------------------------------------------
Search Employee By Eid...
Enter The Employee Eid : 3

Employee Found!

Details...
Eid : 3   Name : Vicky Gupta      Salary : 78900
---------------------------------------------------------



___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 4
---------------------------------------------------------
Search Employee By Name...
Enter The Name Of Your Employee : Anuj Sharma

Employee Found!

Details...
Eid : 2   Name : Anuj Sharma      Salary : 87890
---------------------------------------------------------
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 5
--------------------------------------------------
Highest Salary Of Employee
Eid : 1   Name : Jugnu Gupta     Salary : 98990
--------------------------------------------------



___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 6
--------------------------------------------------

No Of Employee...

Total No Of Employee : 3
--------------------------------------------------
```



```
___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 6
------------------------------------------

No Of Employee...

Total No Of Employee : 3
------------------------------------------



___Operations___
1.Add Employee
2.Display Employee
3.Search Employee Byy Eid
4.Search Employee By Name
5.Employee having Higest Salary
6.Total No Of Employee
7.Exit
Enter Your Choice : 7
------------------------------------------
Exit Operation Is Selected
------------------------------------------

Exiting...
```

Data Structure Lab

CEN-391

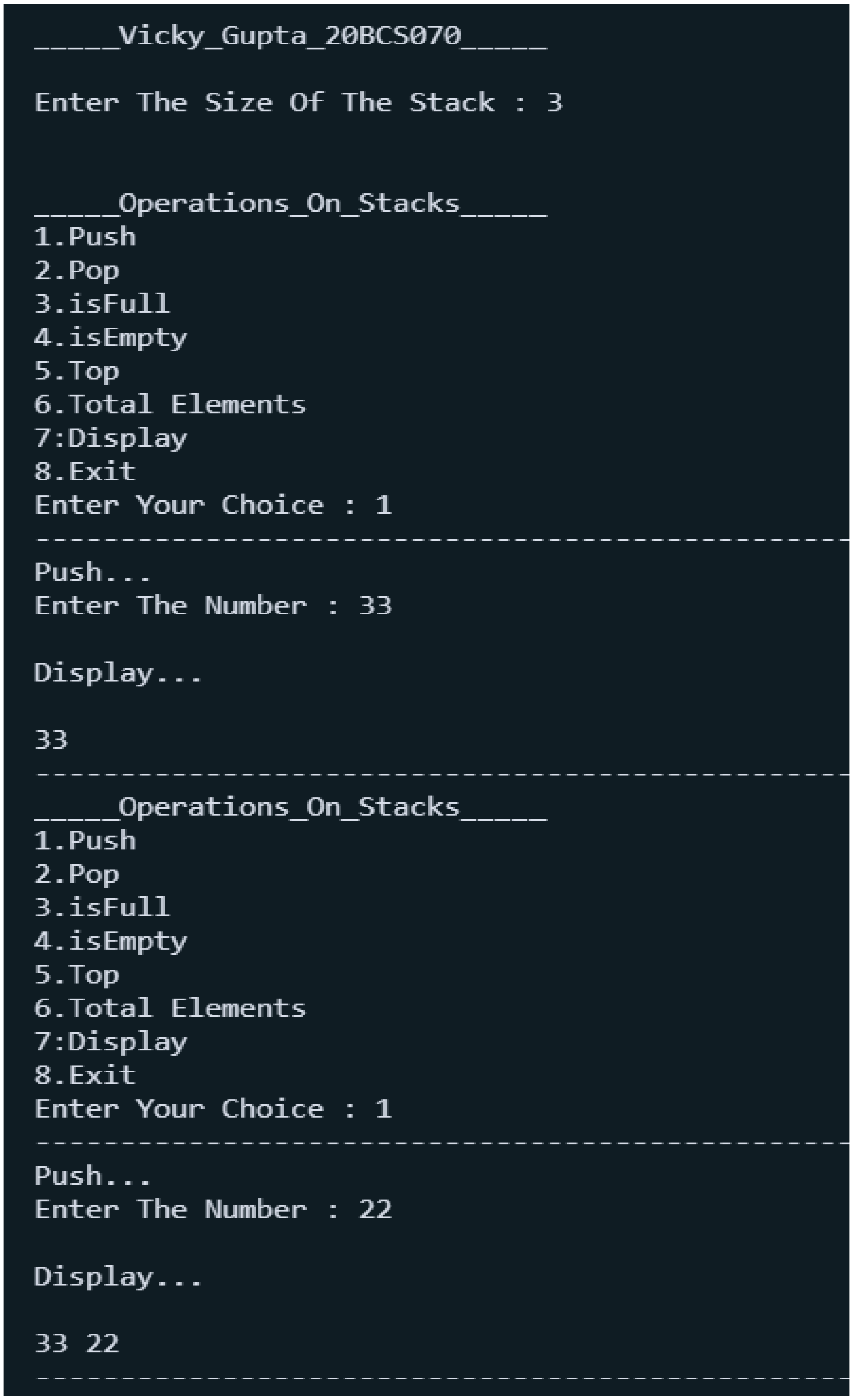
Program 6(a)

Code :-

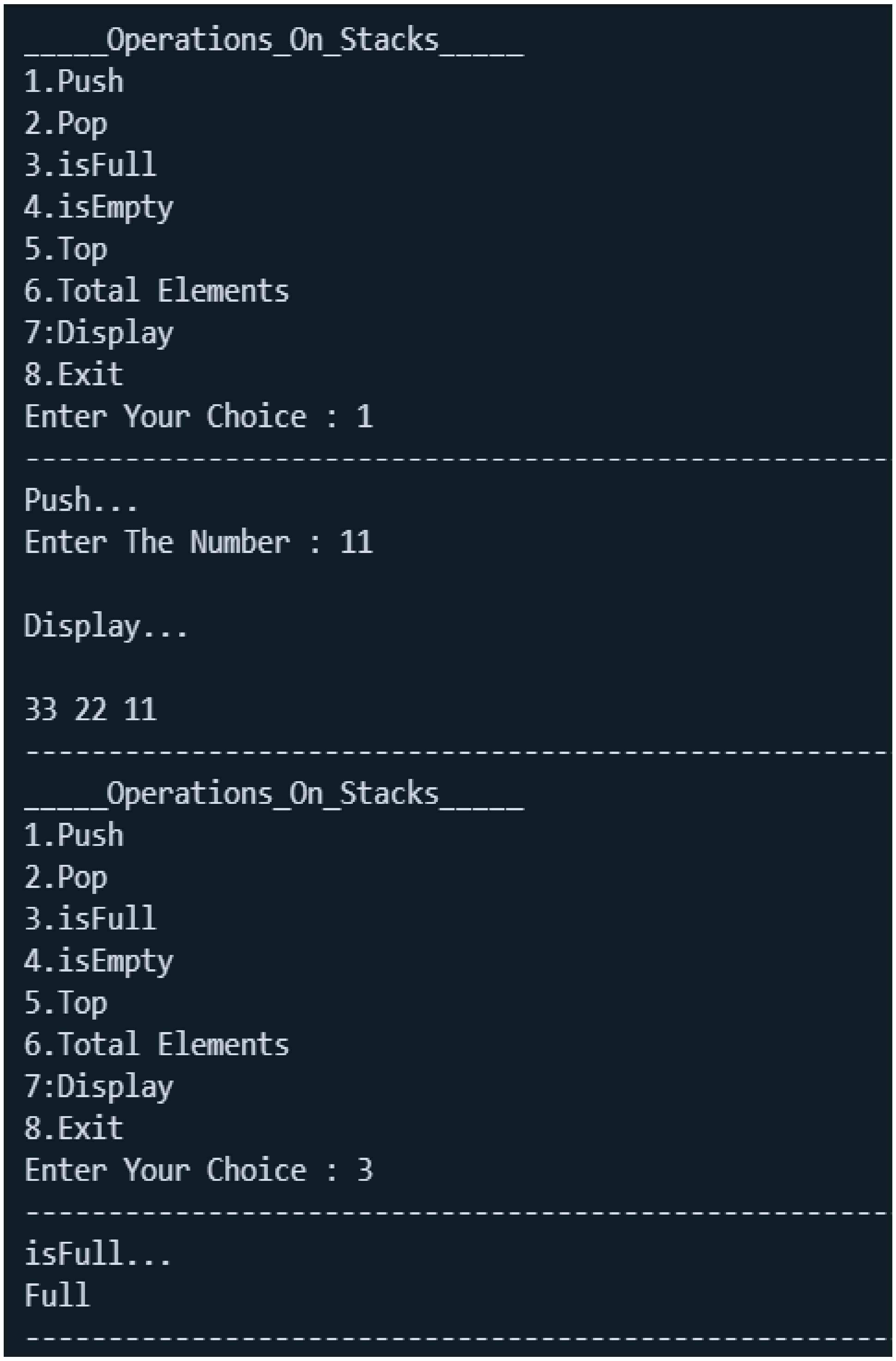
```
#include <iostream>
using namespace std;
int size;
struct stack
{
    int *arr;
    int top;
} st;

void Display()
{
    cout << "Display...\n";
    if (st.top == -1)
    {
        cout << "Stack Is Empty" << endl;
        return;
    }
    cout << "\n";
    for (int i = 0; i <= st.top; i++)
    {
        cout << st.arr[i] << " ";
    }
    cout << "\n";
}

void Push()
{
    cout << "Push...\n";
    if (st.top == size - 1)
    {
        cout << "Stack Overflow" << endl;
        return;
    }
    st.top++;
    int val;
    cout << "Enter The Number : ";
    cin >> val;
    st.arr[st.top] = val;
    cout << "\n";
    Display();
}

void Pop()
{
    cout << "Pop...\n";
    if (st.top == -1)
    {
        cout << "Stack Underflow" << endl;
        return;
    }
    cout << st.arr[st.top] << "\n";
    st.top--;
    cout << "\n";
    Display();
}
```

```
void Top()
{
    cout << "Top...\n";
    if (st.top == -1)
    {
        cout << "Stack Is Empty" << endl;
        return;
    }
    cout << st.arr[st.top] << "\n";
}

void isEmpty()
{
    cout << "isEmpty...\n";
    if (st.top != -1)
    {
        cout << "Not Empty \n";
    }
    else
    {
        cout << "Empty \n";
    }
}

void isFull()
{
    cout << "isFull...\n";
    if (st.top+1 == size)
    {
        cout << "Full \n";
    }
    else
    {
        cout << "Not Full \n";
    }
}

void Total_Elements()
{
    cout << "Total Elements In Stack...\n";
    cout << st.top + 1 << "\n";
}
void Bars()
{
    cout << "---------------------------------------------------
-------------\n";
}
int Options()
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Push();
        break;
    case 2:
        Pop();
        break;
    case 3:
        isFull();
        break;
    case 4:
        isEmpty();
        break;
    case 5:
        Top();
        break;
    case 6:
        Total_Elements();
        break;
    case 7:
        Display();
        break;
    case 8:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "_____Operations_On_Stacks_____ \n";
    cout << "1.Push \n";
    cout << "2.Pop \n";
    cout << "3.isFull \n";
    cout << "4.isEmpty \n";
    cout << "5.Top \n";
    cout << "6.Total Elements \n";
    cout << "7:Display \n";
    cout << "8.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{
    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n";
    cout << "Enter The Size Of The Stack : ";
    cin >> size;
    st.arr = (int *)malloc(size * sizeof(int));
    st.top = -1;
    cout << "\n\n";
    while (true)
    {
        Menu();
        if (!Options())
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

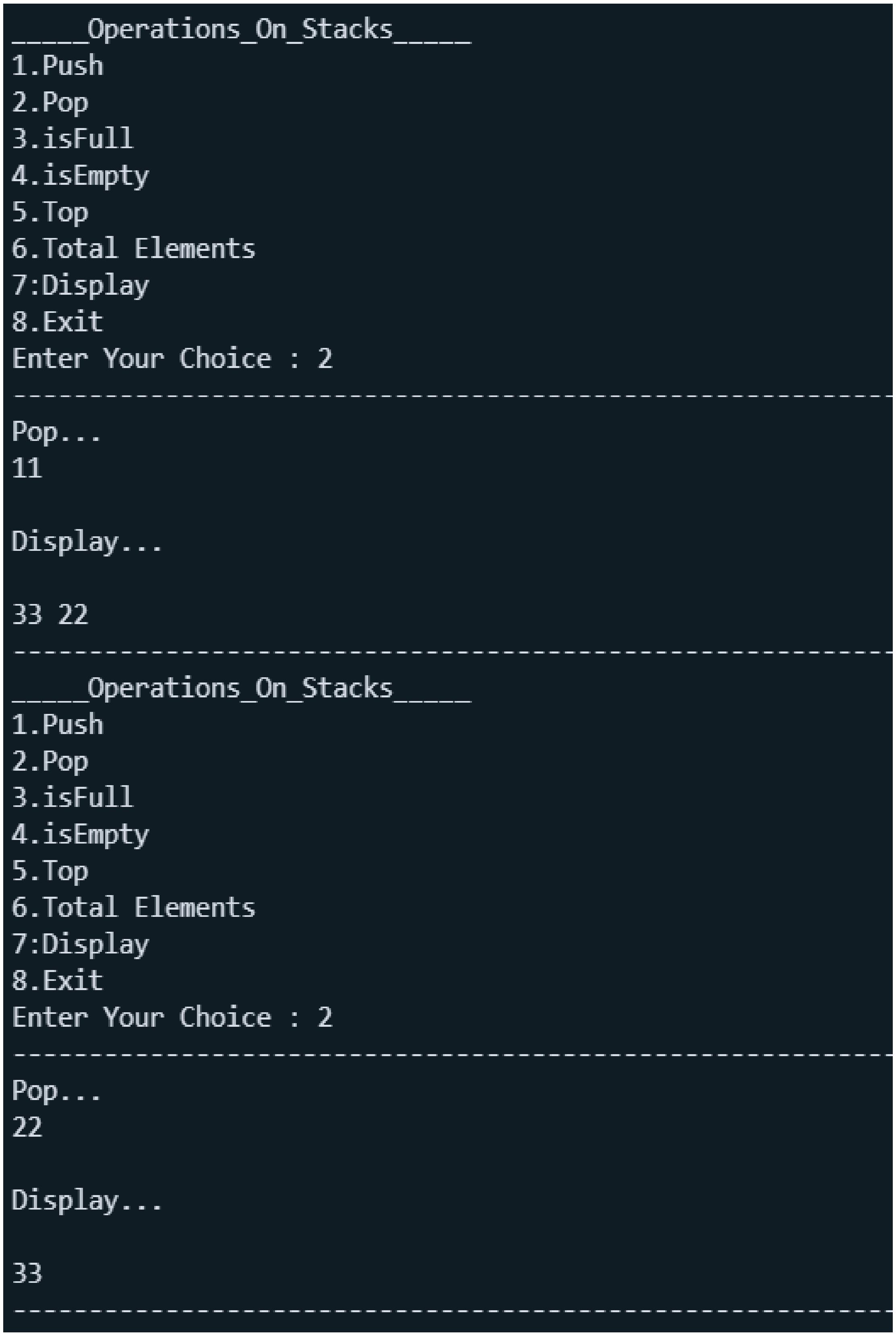
Output :-



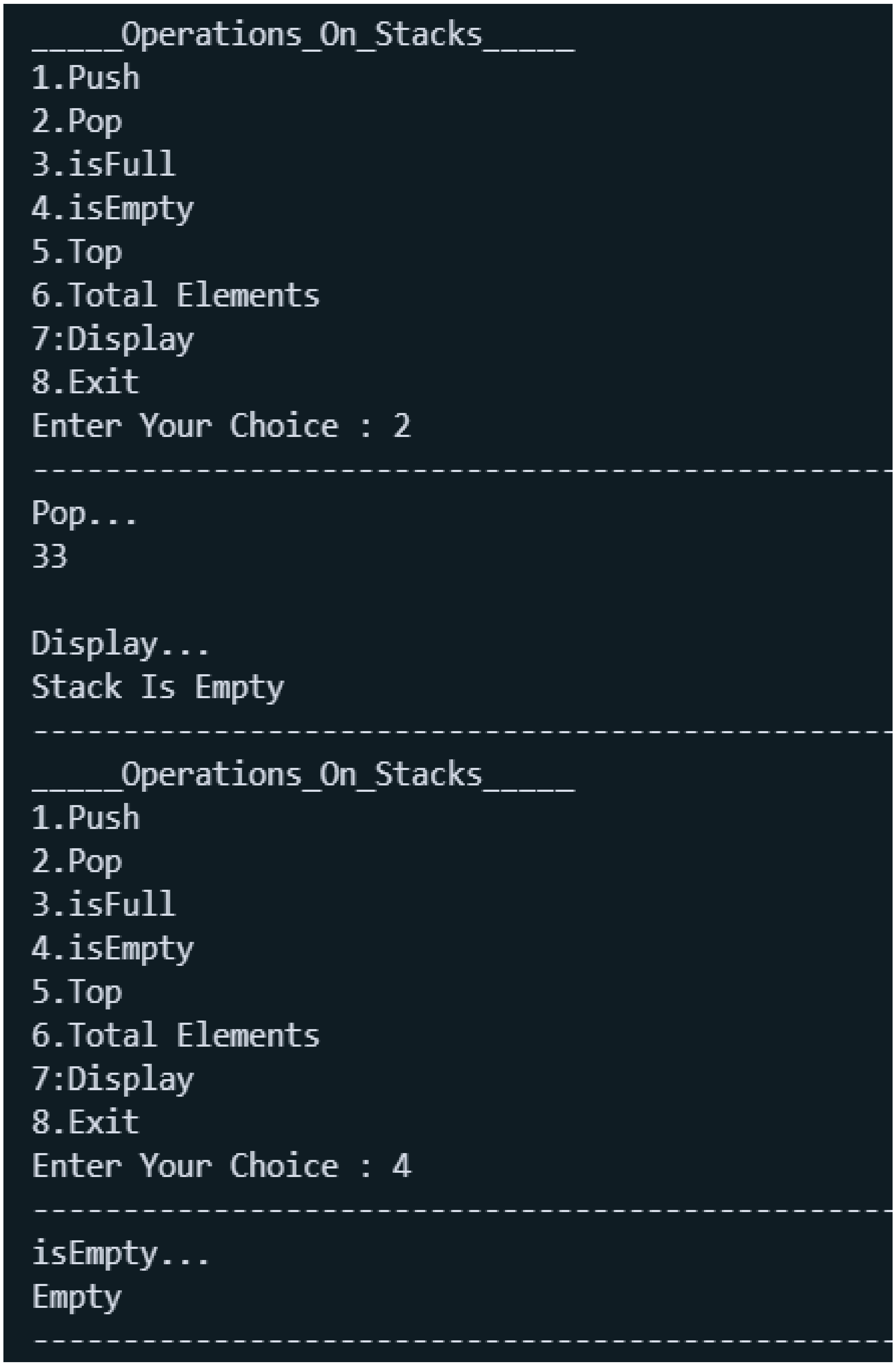
```
_____Vicky_Gupta_20BCS070_____

Enter The Size Of The Stack : 3


_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
------------------------------------------------
Push...
Enter The Number : 33

Display...

33
------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
------------------------------------------------
Push...
Enter The Number : 22

Display...

33 22
------------------------------------------------
```



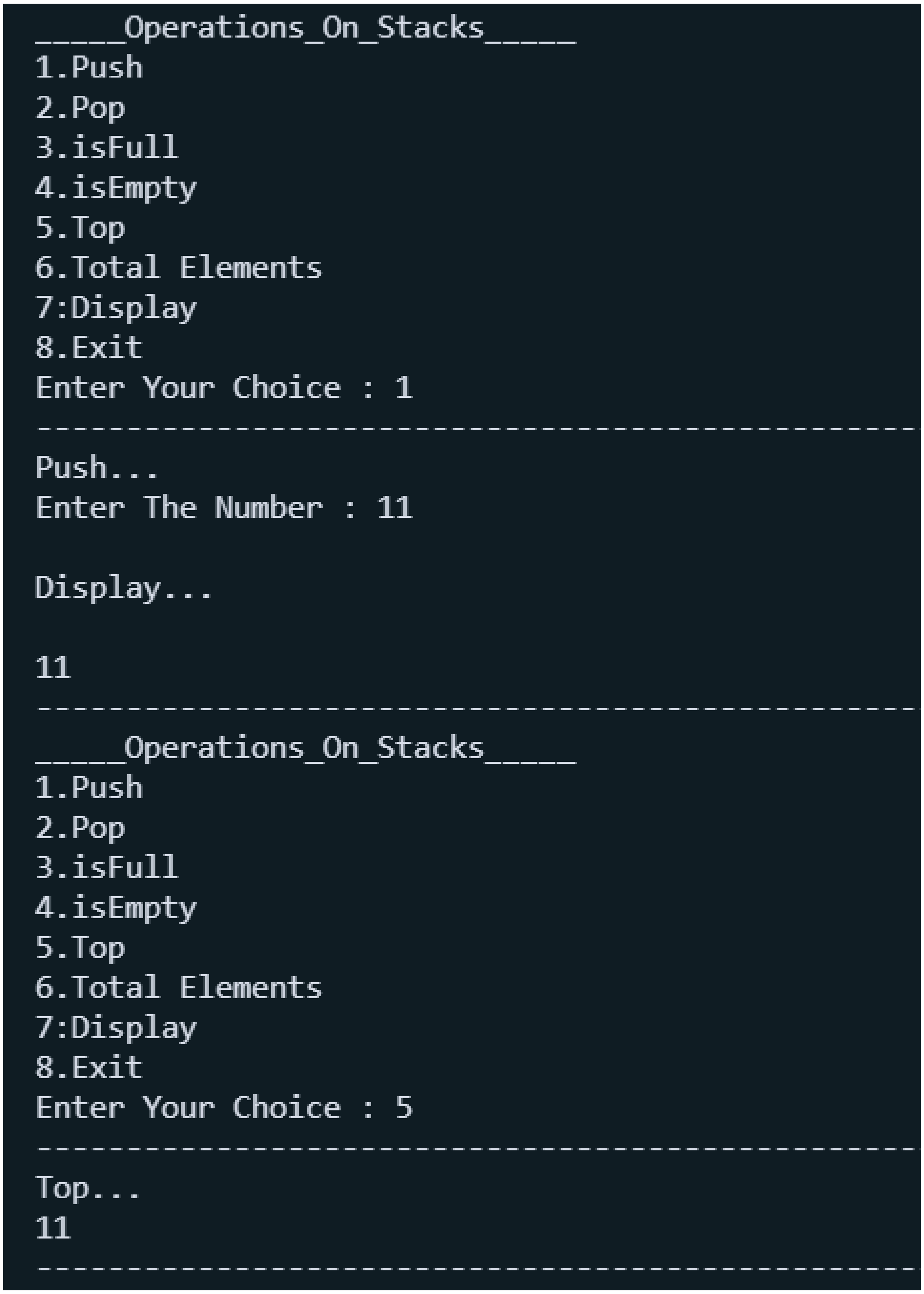
```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
----------------------------------------------------
Push...
Enter The Number : 11

Display...

33 22 11
----------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 3
----------------------------------------------------
isFull...
Full
----------------------------------------------------
```



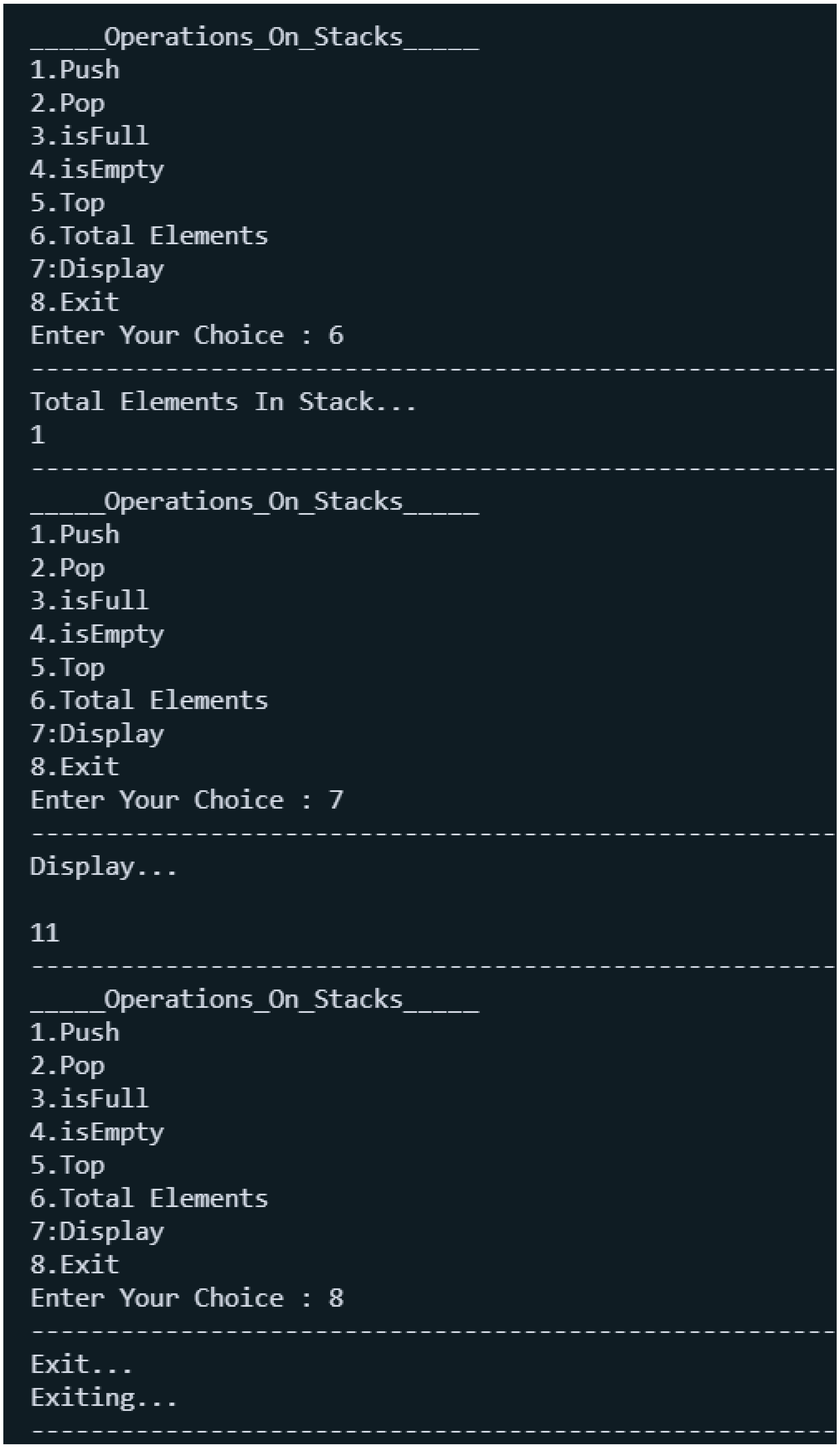
```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------------
Pop...
11

Display...

33 22
----------------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------------
Pop...
22

Display...

33
----------------------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
--------------------------------------------------
Pop...
33

Display...
Stack Is Empty
--------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 4
--------------------------------------------------
isEmpty...
Empty
--------------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
---------------------------------------------------
Push...
Enter The Number : 11

Display...

11
---------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 5
---------------------------------------------------
Top...
11
---------------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 6
-----------------------------------------------------------
Total Elements In Stack...
1
-----------------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 7
-----------------------------------------------------------
Display...

11
-----------------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isFull
4.isEmpty
5.Top
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 8
-----------------------------------------------------------
Exit...
Exiting...
-----------------------------------------------------------
```

Data Structure Lab

CEN-391

Program 6(b)

Code :-

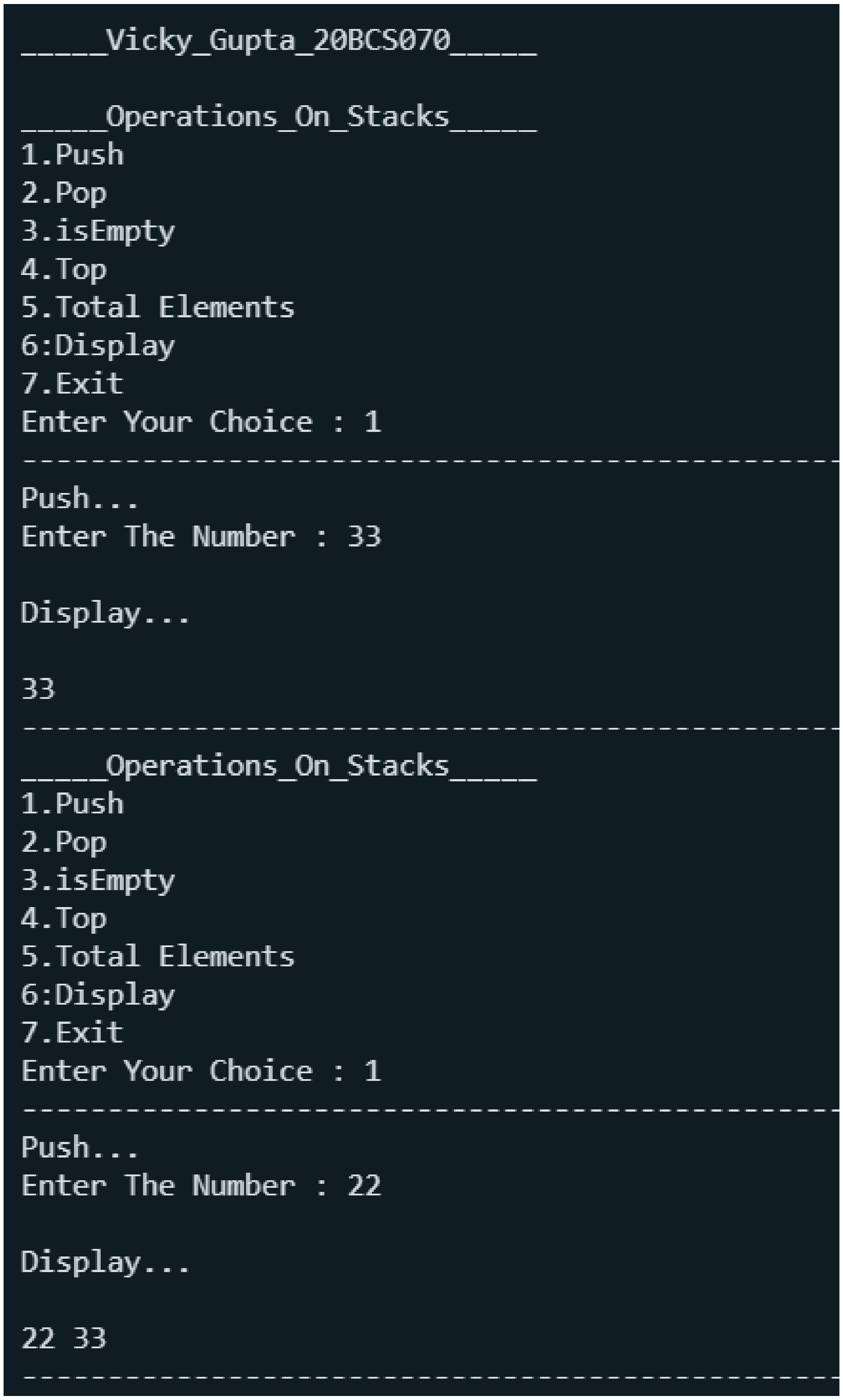
```
#include <iostream>
using namespace std;
struct stack
{
    int data;
    stack *next;
} * top;

void Display()
{
    cout << "Display...\n";
    if (top == nullptr)
    {
        cout << "Stack Is Empty" << endl;
        return;
    }
```

```
        cout << "\n";
        stack *temp = top;
        while (temp != nullptr)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << "\n";
    }

    void Push()
    {
        cout << "Push...\n";
        stack *newnode = (stack *)malloc(sizeof(stack));
        if (newnode == nullptr)
        {
            cout << "Stack Overflow" << endl;
            return;
        }
        cout << "Enter The Number : ";
        cin >> newnode->data;
        newnode->next = top;
        top = newnode;
        cout << "\n";
        Display();
    }

    void Pop()
    {
        cout << "Pop...\n";
        if (top == nullptr)
        {
            cout << "Stack Underflow" << endl;
            return;
        }
        cout << top->data << "\n";
        stack *todelete = top;
        top = top->next;
        delete todelete;
        cout << "\n";
        Display();
    }

    void Top()
    {
        cout << "Top...\n";
        if (top == nullptr)
        {
            cout << "Stack Is Empty" << endl;
            return;
        }
        cout << top->data << "\n";
    }

    void isEmpty()
    {
        cout << "isEmpty...\n";
        if (top != nullptr)
        {
            cout << "Not Empty \n";
        }
        else
        {
            cout << "Empty \n";
        }
    }

    void Total_Elements()
    {
        cout << "Total Elements...\n";
        int total = 0;
        stack *temp = top;
        while (temp != nullptr)
        {
            total++;
            temp = temp->next;
        }
        cout << total << "\n";
    }
    void Bars()
```

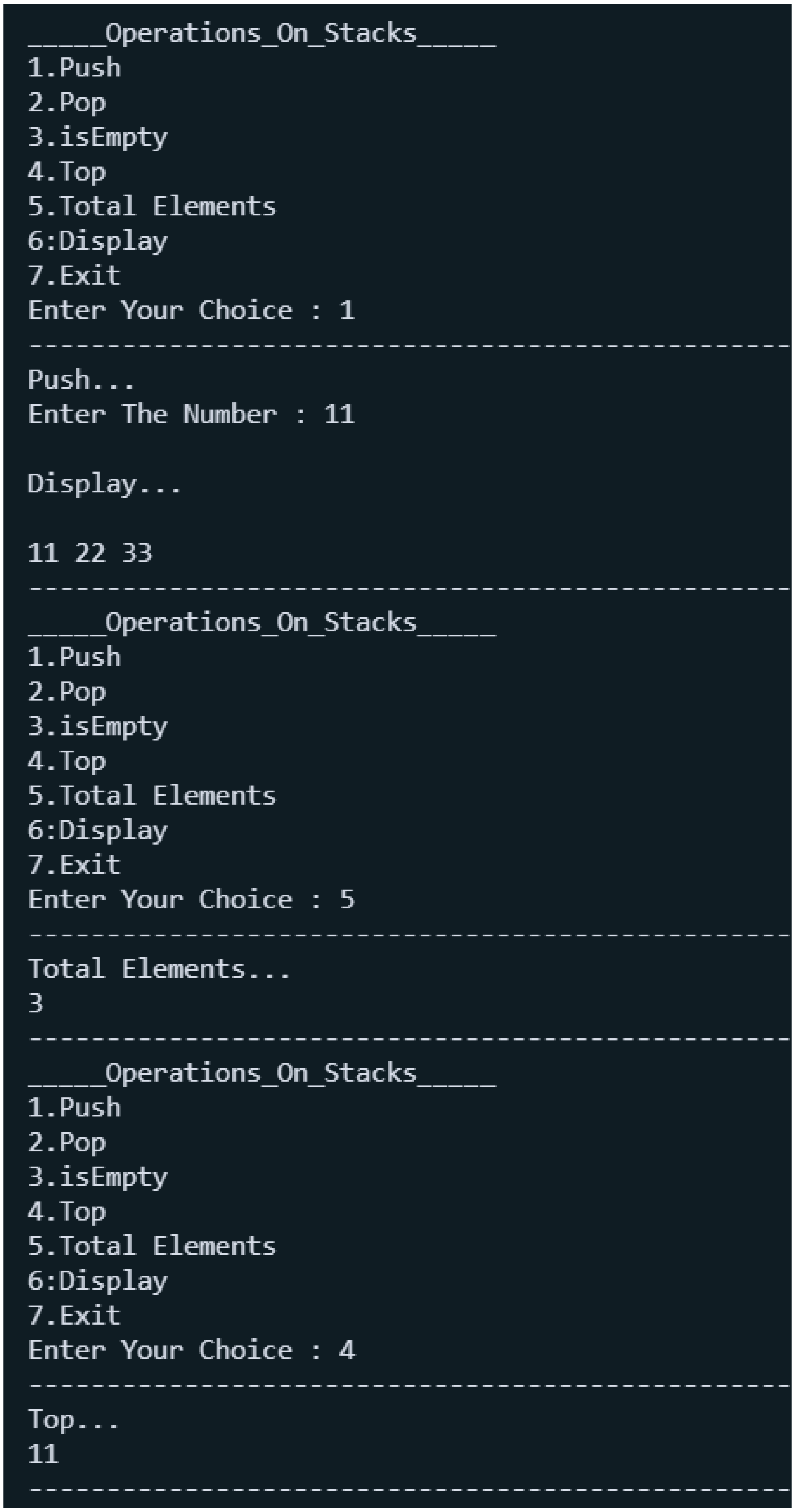
```
{
    cout << "---------------------------------------------------
-------------\n";
}
int Options()
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Push();
        break;
    case 2:
        Pop();
        break;
    case 3:
        isEmpty();
        break;
    case 4:
        Top();
        break;
    case 5:
        Total_Elements();
        break;
    case 6:
        Display();
        break;
    case 7:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
```

```cpp
{
    cout << "_____Operations_On_Stacks_____ \n";
    cout << "1.Push \n";
    cout << "2.Pop \n";
    cout << "3.isEmpty \n";
    cout << "4.Top \n";
    cout << "5.Total Elements \n";
    cout << "6:Display \n";
    cout << "7.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{
    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n\n";

    while (true)
    {
        Menu();
        if (!Options())
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

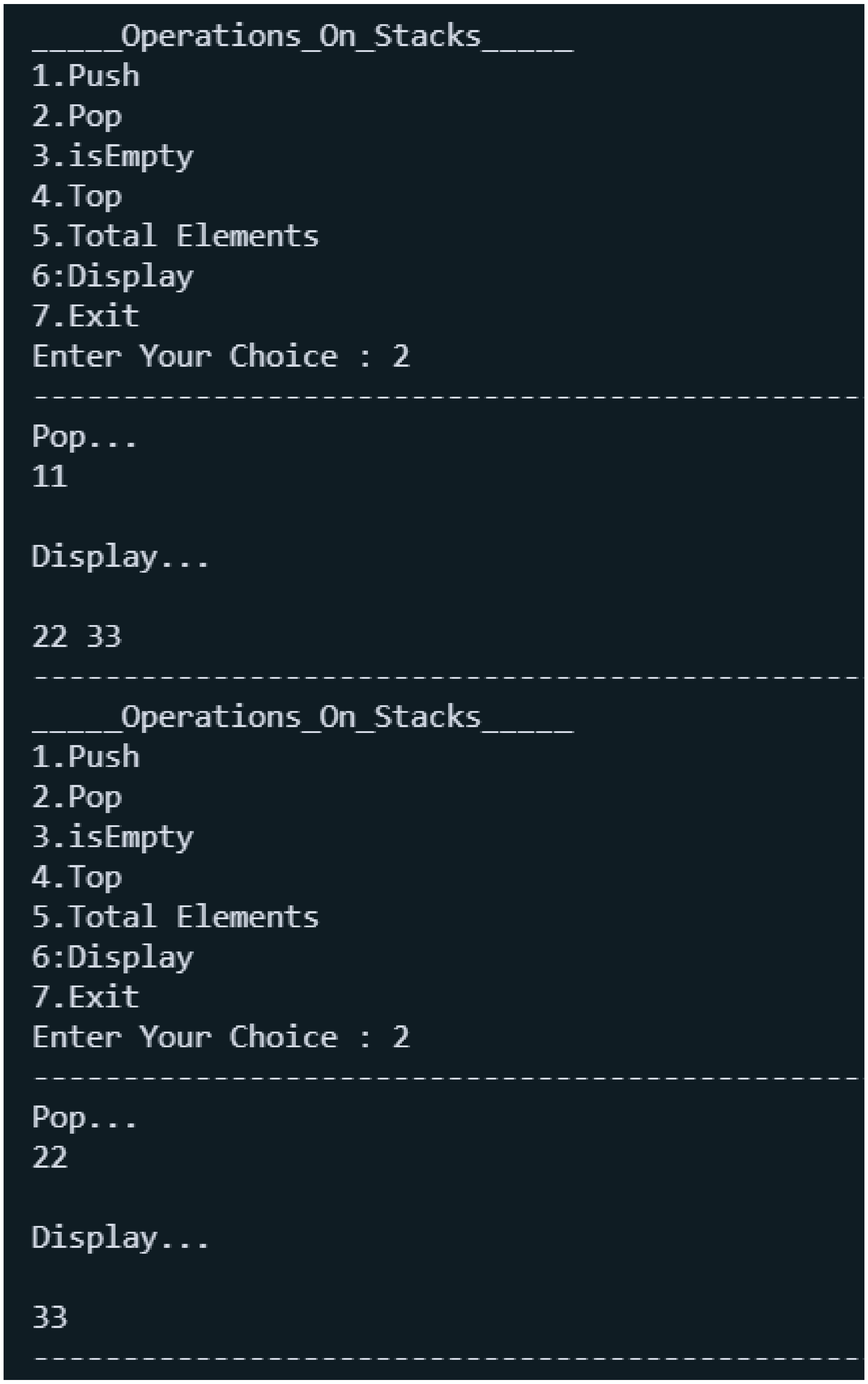
Output :-



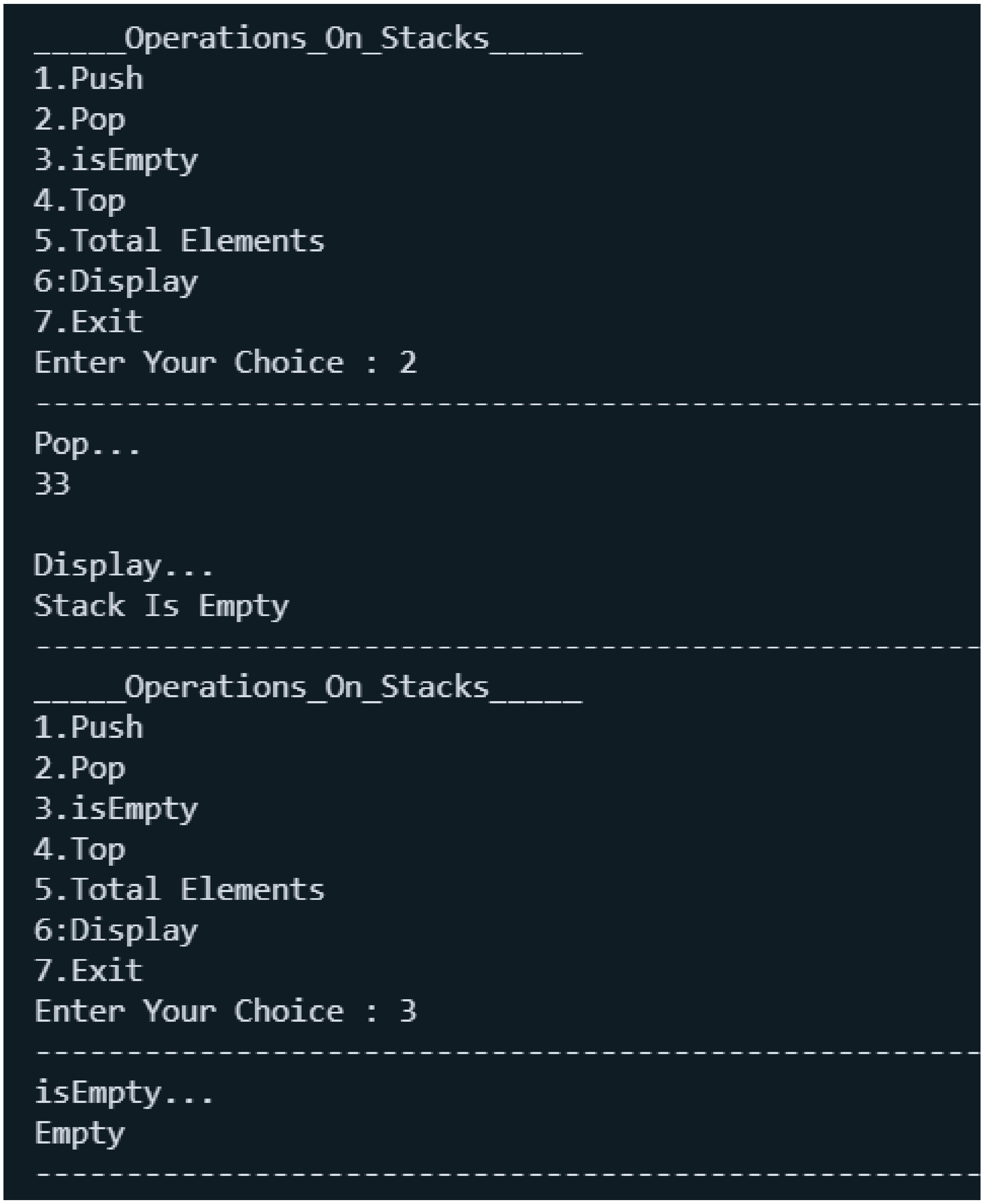
```
_____Vicky_Gupta_20BCS070_____

_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
--------------------------------------------------
Push...
Enter The Number : 33

Display...

33
--------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
--------------------------------------------------
Push...
Enter The Number : 22

Display...

22 33
--------------------------------------------------
```



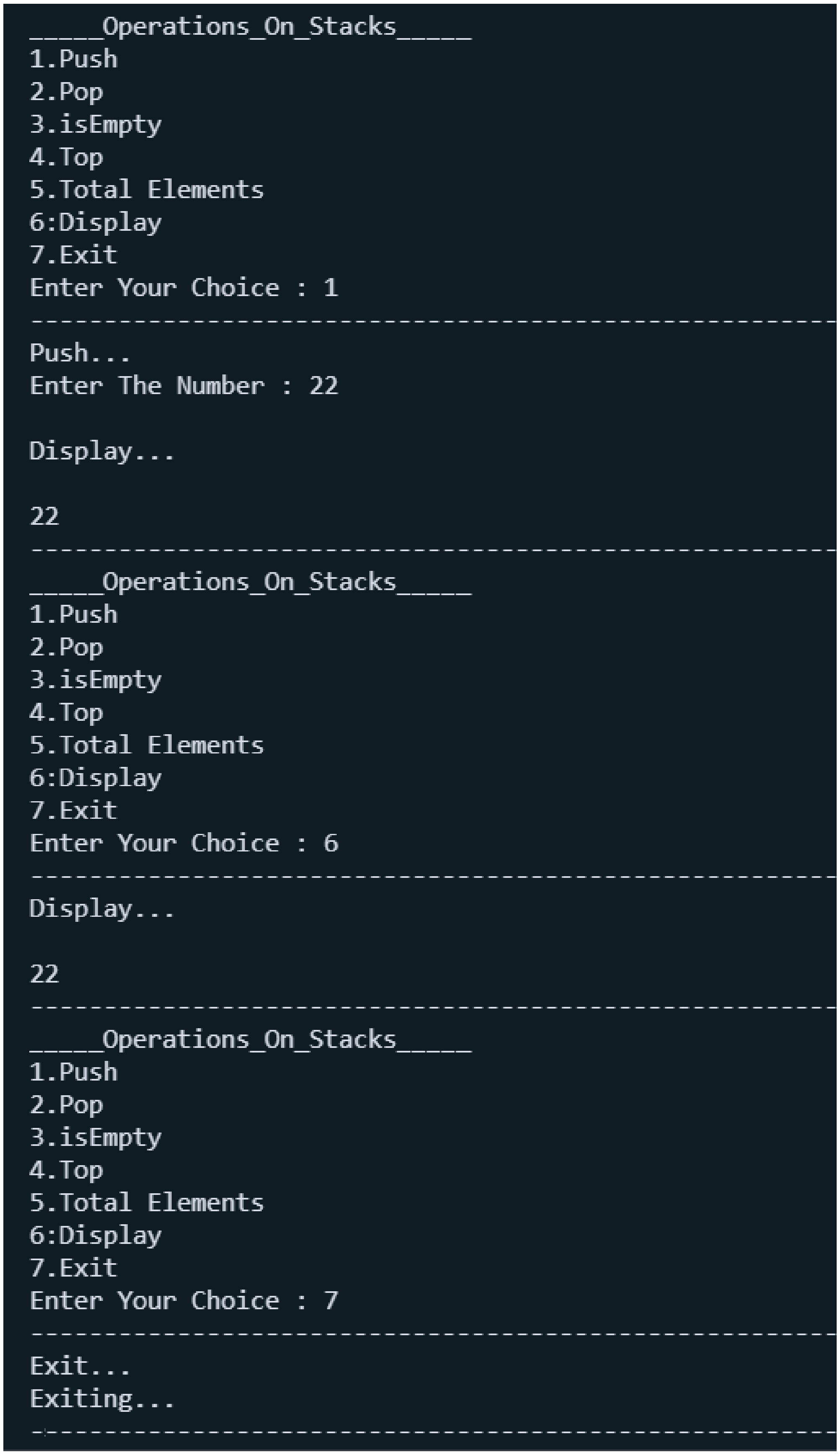
```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
--------------------------------------------------
Push...
Enter The Number : 11

Display...

11 22 33
--------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 5
--------------------------------------------------
Total Elements...
3
--------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 4
--------------------------------------------------
Top...
11
--------------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------------
Pop...
11

Display...

22 33
----------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------------
Pop...
22

Display...

33
----------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------------------
Pop...
33

Display...
Stack Is Empty
----------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 3
----------------------------------------------------
isEmpty...
Empty
----------------------------------------------------
```



```
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
-------------------------------------------------------
Push...
Enter The Number : 22

Display...

22
-------------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 6
-------------------------------------------------------
Display...

22
-------------------------------------------------------
_____Operations_On_Stacks_____
1.Push
2.Pop
3.isEmpty
4.Top
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 7
-------------------------------------------------------
Exit...
Exiting...
-------------------------------------------------------
```

Data Structure Lab

CEN-391

Program 7

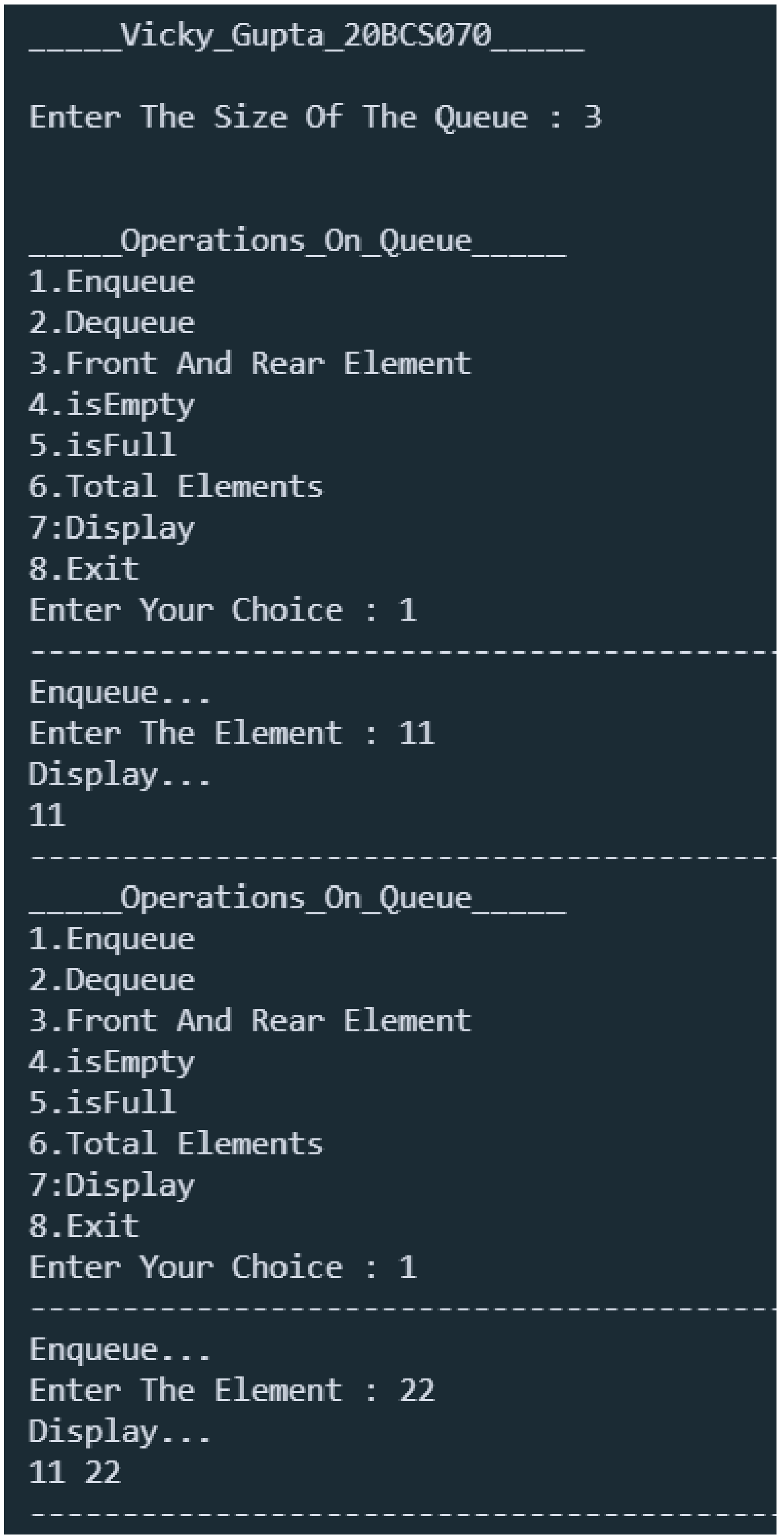
Code :-

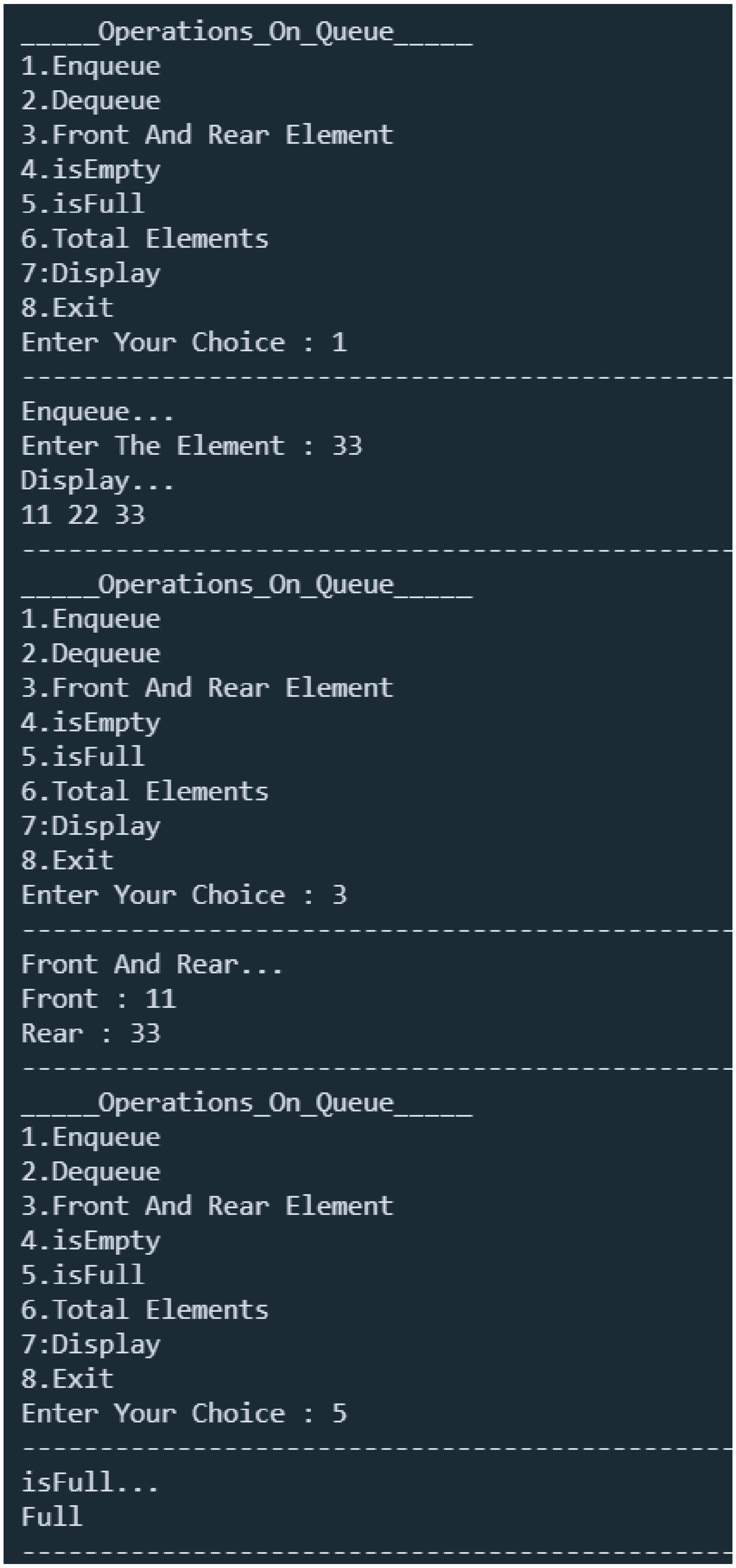
```cpp
#include <iostream>
using namespace std;

void isEmpty(int size)
{
    cout << "isEmpty...\n";
    if (size == -1)
        cout << "Empty" << endl;
    else
        cout << "Not Empty" << endl;
}

void isFull(int size, int capacity)
{
    cout << "isFull...\n";
    if (size + 1 == capacity)
```

```
            cout << "Full" << endl;
        else
            cout << "Not Full" << endl;
    }

    void Display(int queue[], int size)
    {
        cout << "Display...\n";
        if (size == -1)
        {
            cout << "Queue Is Empty" << endl;
            return;
        }
        for (int i = 0; i <= size; i++)
        {
            cout << queue[i] << " ";
        }
        cout << endl;
    }

    void Enqueue(int queue[], int &size, int capacity)
    {
        cout << "Enqueue...\n";
        size++;
        if (size == capacity)
        {
            size--;
            cout << "Queue Overflow" << endl;
            return;
        }
        cout << "Enter The Element : ";
        cin >> queue[size];
        Display(queue, size);
    }

    void Dequeue(int queue[], int &size)
    {
        cout << "Dequeue...\n";
        if (size == -1)
        {
            cout << "Queue Underflow" << endl;
            return;
        }
        cout<<queue[0]<<endl;
        for (int i = 1; i <= size; i++)
        {
            queue[i - 1] = queue[i];
        }
        size--;
        Display(queue, size);
    }
    void Front_Rear(int queue[], int size)
    {
        cout << "Front And Rear...\n";
        if (size == -1)
        {
            cout << "Queue Is Empty" << endl;
            return;
        }
        cout << "Front : " << queue[0] << endl;
        cout << "Rear : " << queue[size] << endl;
    }

    void Total_Element(int size)
    {
        cout << "Total Elements In Queue : " << size + 1 <<
    endl;
    }

    void Bars()
    {
        cout << "------------------------------------------------
    ----------------\n";
    }
    bool Options(int queue[], int &size, int capacity)
    {
        int opt;
        cin >> opt;
```

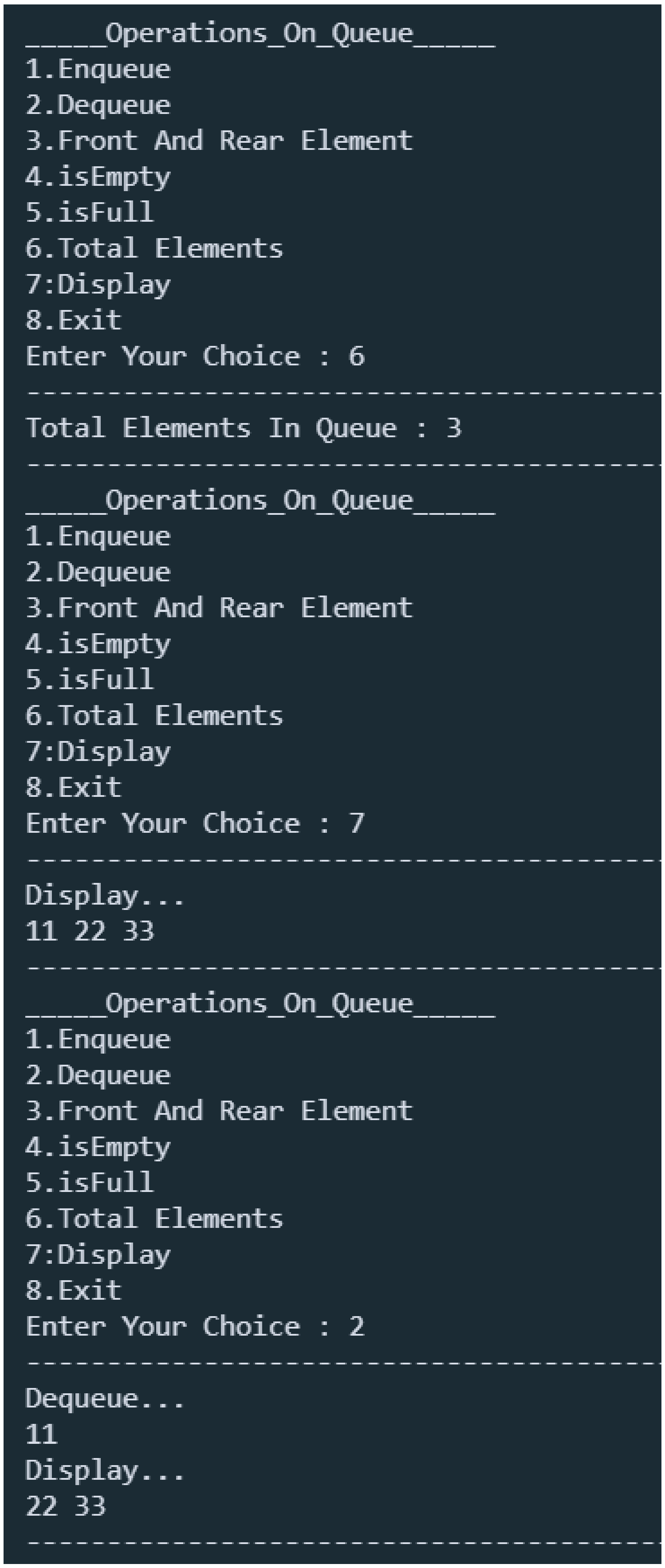
```
        Bars();
        switch (opt)
        {
        case 1:
            Enqueue(queue, size, capacity);
            break;
        case 2:
            Dequeue(queue, size);
            break;
        case 3:
            Front_Rear(queue, size);
            break;
        case 4:
            isEmpty(size);
            break;
        case 5:
            isFull(size, capacity);
            break;
        case 6:
            Total_Element(size);
            break;
        case 7:
            Display(queue, size);
            break;
        case 8:
            cout << "Exit...\n";
            return 0;
        default:
            cout << "Invalid Input!\nTry Again!\n";
        }
        Bars();
        return 1;
    }

    void Menu()
    {
        cout << "_____Operations_On_Queue_____ \n";
        cout << "1.Enqueue \n";
        cout << "2.Dequeue \n";
        cout << "3.Front And Rear Element \n";
        cout << "4.isEmpty \n";
        cout << "5.isFull \n";
        cout << "6.Total Elements \n";
        cout << "7:Display \n";
        cout << "8.Exit \n";
        cout << "Enter Your Choice : ";
    }

    int main()
    {

        system("cls");
        cout << "_____Vicky_Gupta_20BCS070_____\n\n";
        cout << "Enter The Size Of The Queue : ";
        int capacity, size = -1;
        cin >> capacity;
        int queue[capacity] = {0};
        cout << "\n\n";
        while (true)
        {
            Menu();
            if (!Options(queue, size, capacity))
                break;
        }
        cout << "Exiting...\n";
        Bars();
        return 0;
    }
```

Output :-

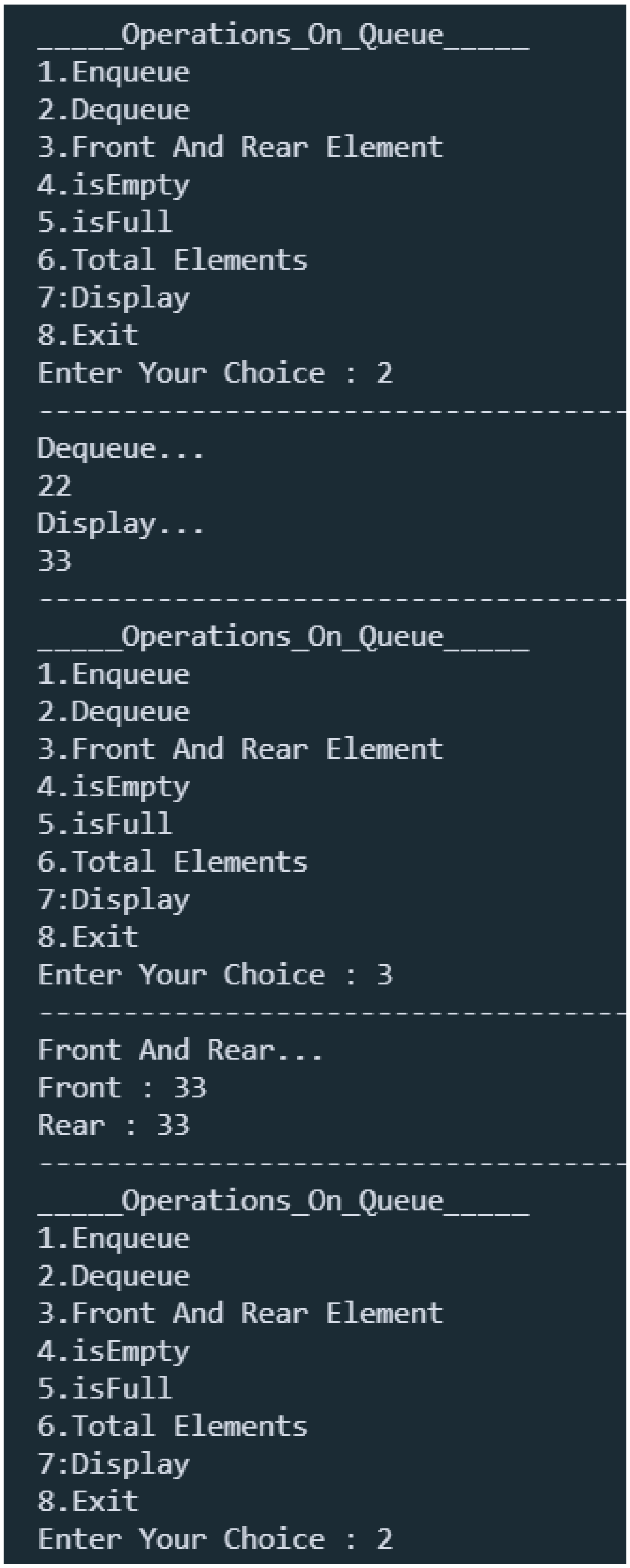

```
_____Vicky_Gupta_20BCS070_____

Enter The Size Of The Queue : 3


_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
----------------------------------------
Enqueue...
Enter The Element : 11
Display...
11
----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
----------------------------------------
Enqueue...
Enter The Element : 22
Display...
11 22
----------------------------------------
```



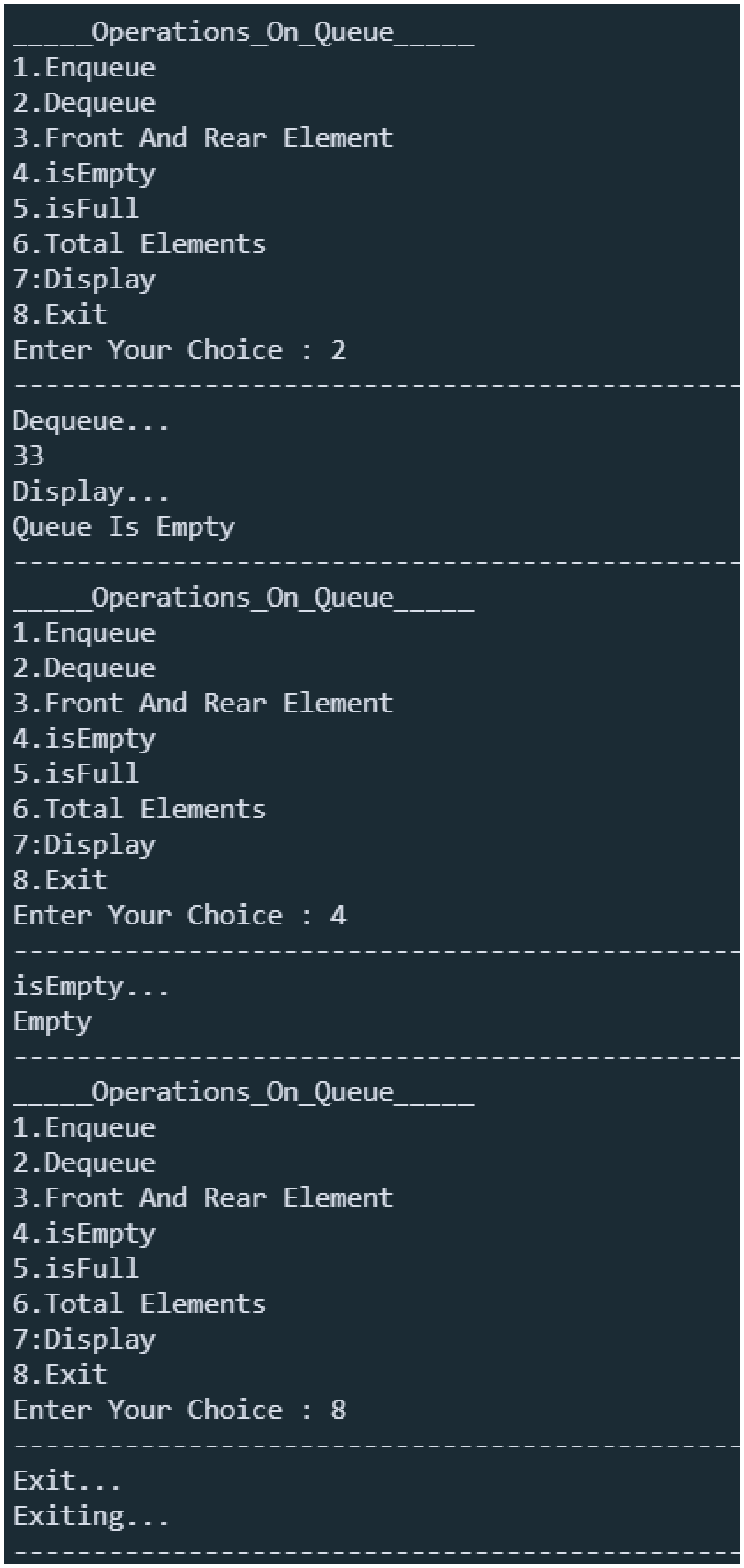
```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
----------------------------------------------
Enqueue...
Enter The Element : 33
Display...
11 22 33
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 3
----------------------------------------------
Front And Rear...
Front : 11
Rear : 33
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 5
----------------------------------------------
isFull...
Full
----------------------------------------------
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 6
----------------------------------------
Total Elements In Queue : 3
----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 7
----------------------------------------
Display...
11 22 33
----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------
Dequeue...
11
Display...
22 33
----------------------------------------
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
-----------------------------------
Dequeue...
22
Display...
33
-----------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 3
-----------------------------------
Front And Rear...
Front : 33
Rear : 33
-----------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------
Dequeue...
33
Display...
Queue Is Empty
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 4
----------------------------------------------
isEmpty...
Empty
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 8
----------------------------------------------
Exit...
Exiting...
----------------------------------------------
```

Data Structure Lab

CEN-391

Program 8

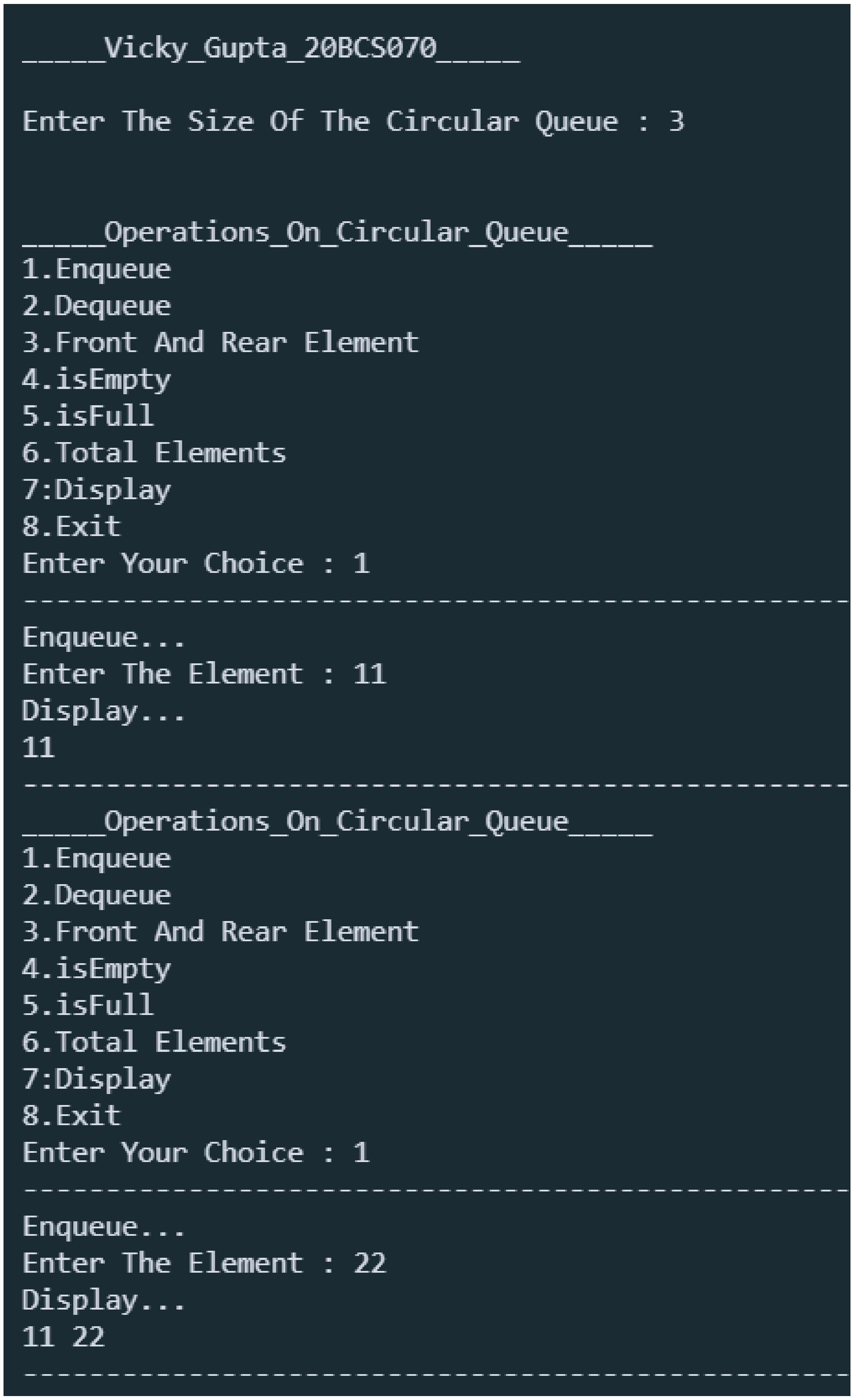
Code :-

```cpp
#include <iostream>
using namespace std;

void isEmpty(int front, int rear)
{
    cout << "isEmpty...\n";
    if (front == -1 && rear == -1)
        cout << "Empty" << endl;
    else
        cout << "Not Empty" << endl;
}

void isFull(int front, int rear, int capacity)
{
    cout << "isFull...\n";
    if ((rear + 1) % capacity == front)
```

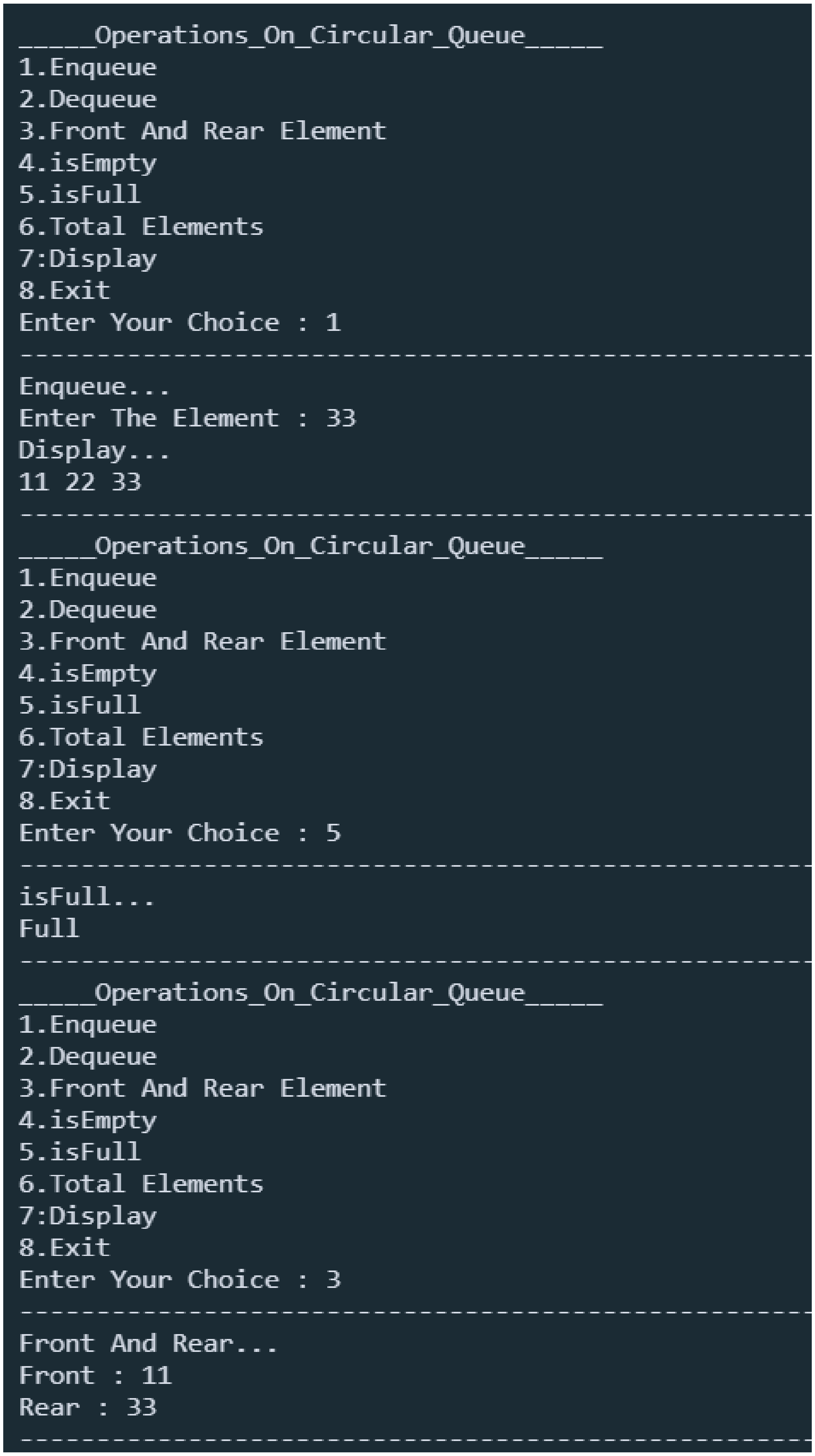
```
        cout << "Full" << endl;
    else
        cout << "Not Full" << endl;
}

void Display(int queue[], int front, int rear, int capacity)
{
    cout << "Display...\n";
    if (rear == -1 && front == -1)
    {
        cout << "Queue Empty" << endl;
        return;
    }
    if (front <= rear)
    {
        for (int i = front; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }
    else
    {
        for (int i = front; i < capacity; i++)
        {
            cout << queue[i] << " ";
        }
        for (int i = 0; i <= rear; i++)
        {
            cout << queue[i] << " ";
        }
    }

    cout << endl;
}

void Enqueue(int queue[], int &front, int &rear, int
capacity)
{
    cout << "Enqueue...\n";
    if (front == -1 && rear == -1)
    {
        front = 0;
        rear = 0;
        cout << "Enter The Element : ";
        cin >> queue[rear];
        Display(queue, front, rear, capacity);
    }
    else if ((rear + 1) % capacity == front)
    {
        cout << "Queue Overflow" << endl;
    }
    else
    {
        rear = (rear + 1) % capacity;
        cout << "Enter The Element : ";
        cin >> queue[rear];
        Display(queue, front, rear, capacity);
    }
}

void Dequeue(int queue[], int &front, int &rear, int
capacity)
{
    cout << "Dequeue...\n";
    if (rear == -1 && front == -1)
    {
        cout << "Queue Underflow" << endl;
    }
    else if (front == rear)
    {
        cout << queue[front] << endl;
        front = -1;
        rear = -1;
        Display(queue, front, rear, capacity);
    }
    else
    {
        cout << queue[front] << endl;
        front = (front + 1) % capacity;
        Display(queue, front, rear, capacity);
    }
}
void Front_Rear(int queue[], int front, int rear)
{
    cout << "Front And Rear...\n";
    if (front == -1 && rear == -1)
    {
        cout << "Queue Is Empty" << endl;
    }
    cout << "Front : " << queue[front] << endl;
    cout << "Rear : " << queue[rear] << endl;
}

void Total_Element(int front, int rear, int capacity)
{
    if (front == -1 && rear == -1)
        cout << "Total Elements In Queue : " << 0 << endl;
    else if (front <= rear)
        cout << "Total Elements In Queue : " << rear - front
+ 1 << endl;
    else
        cout << "Total Elements In Queue : " << front -
capacity + rear + 1 << endl;
}

void Bars()
{
    cout << "------------------------------------------------
----------------\n";
}
bool Options(int queue[], int &front, int &rear, int
capacity)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
```

```cpp
    {
    case 1:
        Enqueue(queue, front, rear, capacity);
        break;
    case 2:
        Dequeue(queue, front, rear, capacity);
        break;
    case 3:
        Front_Rear(queue, front, rear);
        break;
    case 4:
        isEmpty(front, rear);
        break;
    case 5:
        isFull(front, rear, capacity);
        break;
    case 6:
        Total_Element(front, rear, capacity);
        break;
    case 7:
        Display(queue, front, rear, capacity);
        break;
    case 8:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "_____Operations_On_Circular_Queue_____ \n";
    cout << "1.Enqueue \n";
    cout << "2.Dequeue \n";
    cout << "3.Front And Rear Element \n";
    cout << "4.isEmpty \n";
    cout << "5.isFull \n";
    cout << "6.Total Elements \n";
    cout << "7:Display \n";
    cout << "8.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{

    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n\n";
    cout << "Enter The Size Of The Circular Queue : ";
    int capacity, front = -1, rear = -1;
    cin >> capacity;
    int *queue = (int *)malloc(sizeof(int) * capacity);
    cout << "\n\n";
    while (true)
    {
        Menu();
        if (!Options(queue, front, rear, capacity))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

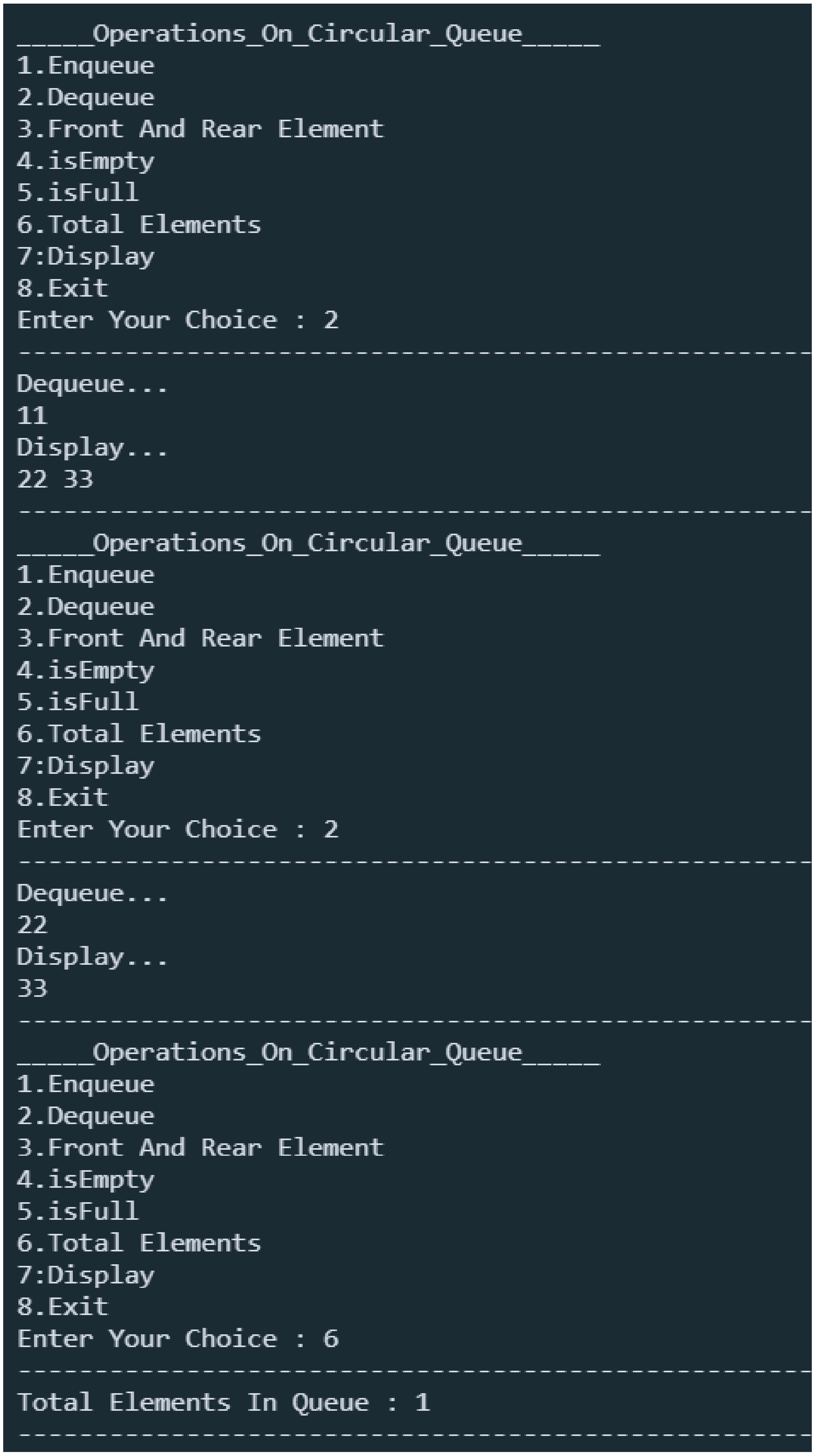
Output :-

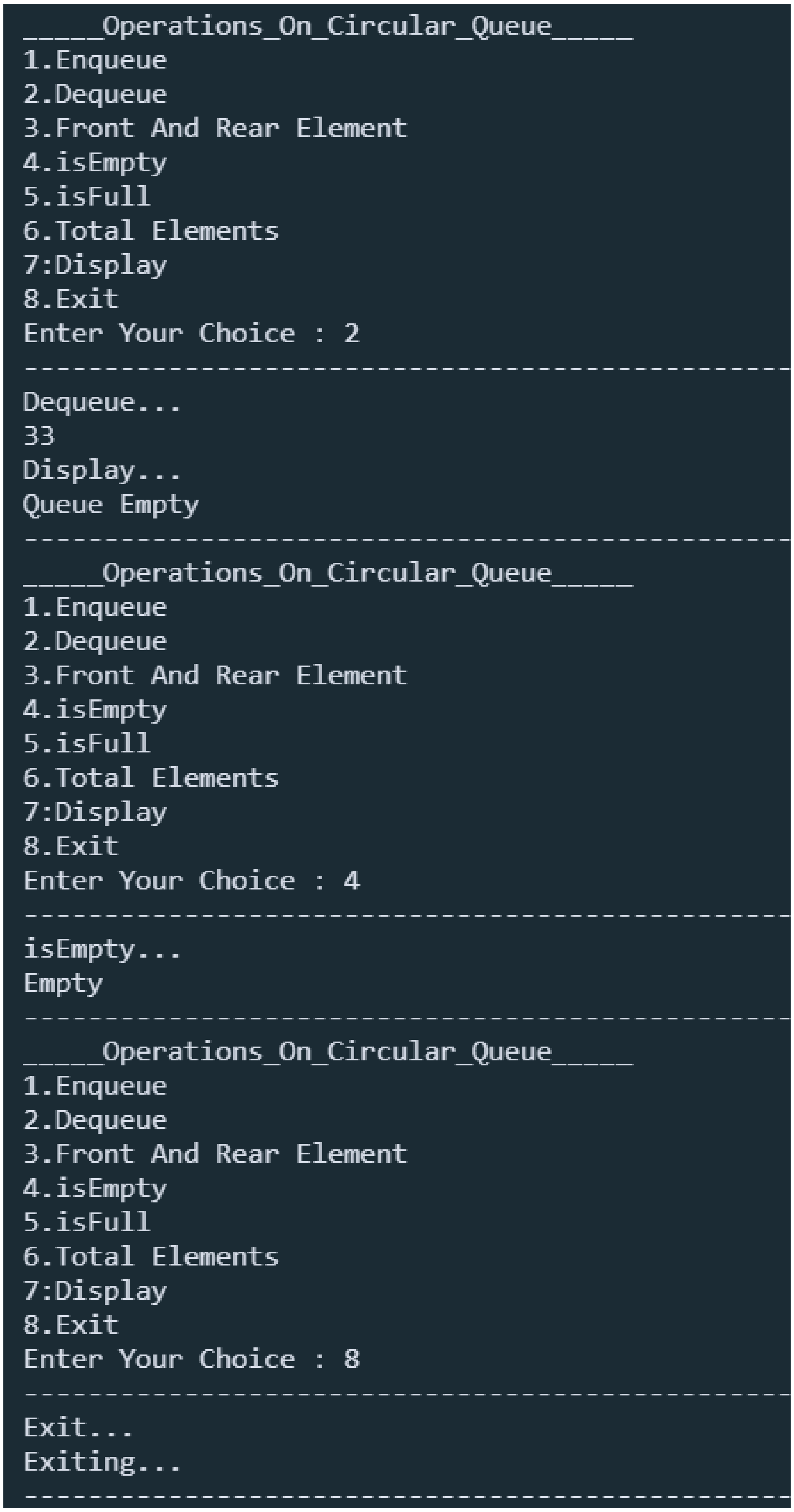


```
_____Vicky_Gupta_20BCS070_____

Enter The Size Of The Circular Queue : 3


_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
--------------------------------------------------
Enqueue...
Enter The Element : 11
Display...
11
--------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
--------------------------------------------------
Enqueue...
Enter The Element : 22
Display...
11 22
--------------------------------------------------
```



```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 1
---------------------------------------------------
Enqueue...
Enter The Element : 33
Display...
11 22 33
---------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 5
---------------------------------------------------
isFull...
Full
---------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 3
---------------------------------------------------
Front And Rear...
Front : 11
Rear : 33
---------------------------------------------------
```


```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------
Dequeue...
11
Display...
22 33
----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
----------------------------------------------------
Dequeue...
22
Display...
33
----------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 6
----------------------------------------------------
Total Elements In Queue : 1
----------------------------------------------------
```



```
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 2
------------------------------------------------
Dequeue...
33
Display...
Queue Empty
------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 4
------------------------------------------------
isEmpty...
Empty
------------------------------------------------
_____Operations_On_Circular_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.isFull
6.Total Elements
7:Display
8.Exit
Enter Your Choice : 8
------------------------------------------------
Exit...
Exiting...
------------------------------------------------
```

Data Structure Lab

CEN-391

Program 9

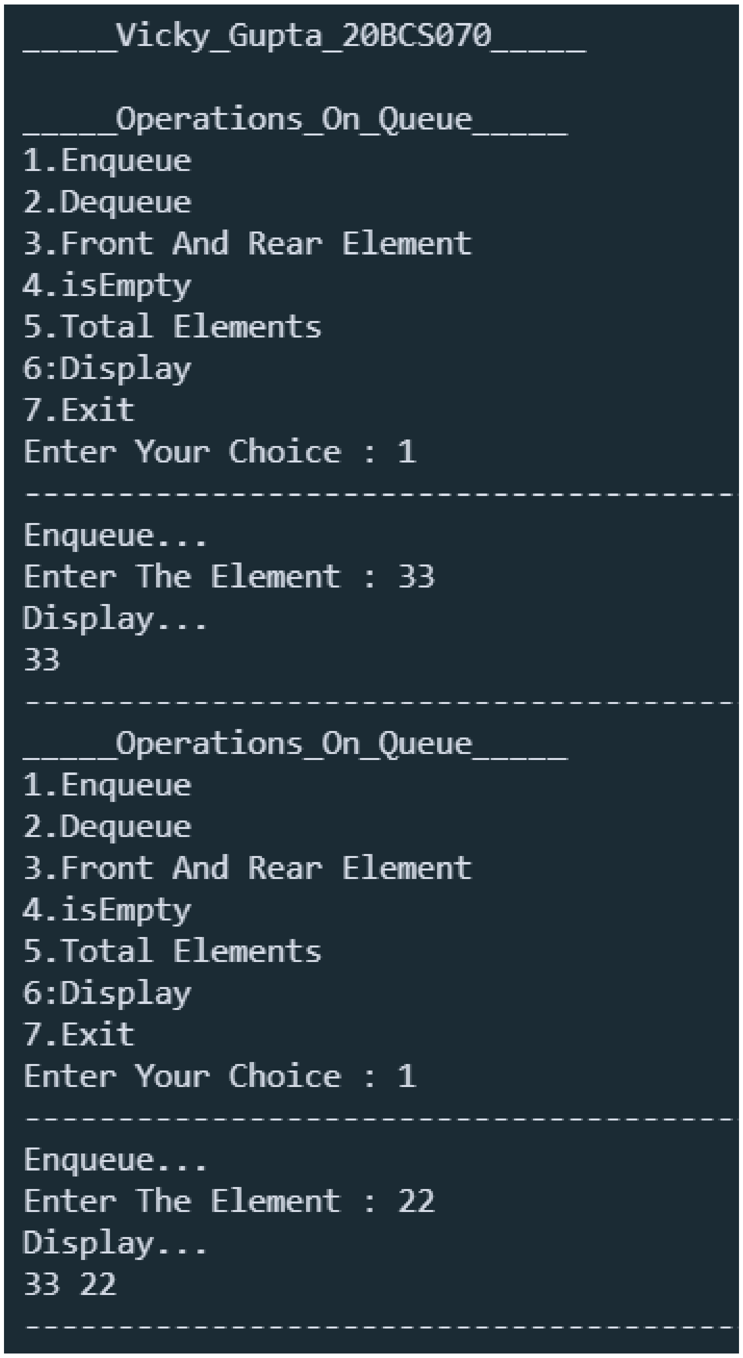
Code :-

```
#include <iostream>
using namespace std;

struct Node
{
    int data;
    Node *next;
};

void
isEmpty(int size)
{
    cout << "isEmpty...\n";
    if (size == 0)
        cout << "Empty" << endl;
    else
```

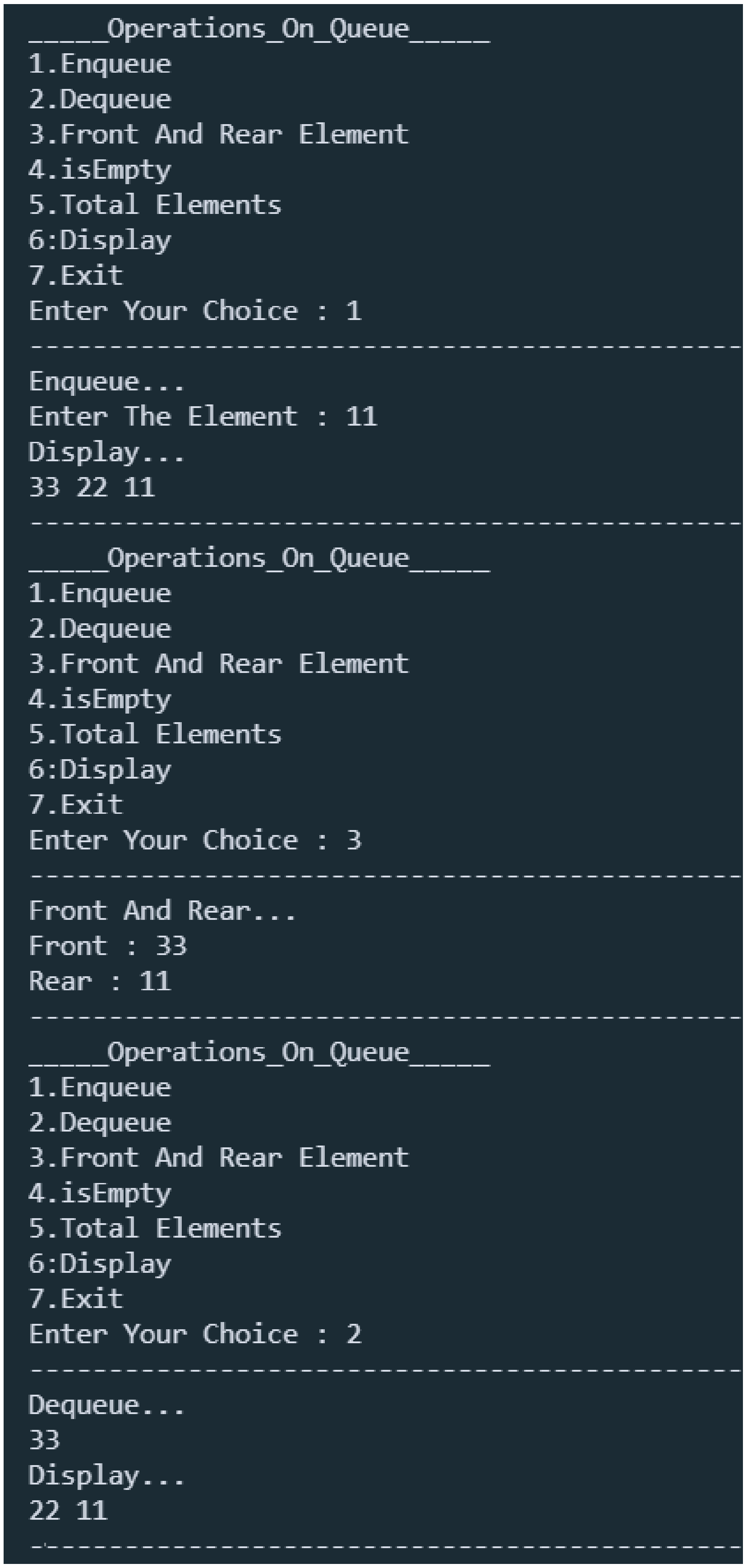
```
        cout << "Not Empty" << endl;
}

void Display(Node *head, int size)
{
    cout << "Display...\n";
    if (size == 0)
    {
        cout << "Queue Is Empty" << endl;
        return;
    }
    while (head != nullptr)
    {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

void Enqueue(Node *&head, Node *&tail, int &size)
{
    cout << "Enqueue...\n";
    size++;
    Node *newnode = (Node *)malloc(1 * sizeof(Node));
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned" << endl;
        return;
    }
    cout << "Enter The Element : ";
    int val;
    cin >> val;
    newnode->data = val;
    newnode->next=nullptr;
    if (head != nullptr)
    {
        tail->next = newnode;
        tail = tail->next;
    }
    else
    {
        head = newnode;
        tail = newnode;
    }
    Display(head, size);
}

void Dequeue(Node *&head, int &size)
{
    cout << "Dequeue...\n";
    if (size == 0)
    {
        cout << "Queue Underflow" << endl;
        return;
    }
    cout << head->data << endl;
    size--;
    Node *todelete = head;
    head = head->next;
    delete todelete;
    Display(head, size);
}
void Front_Rear(Node *head, Node *tail, int size)
{
    cout << "Front And Rear...\n";
    if (size == 0)
    {
        cout << "Queue Is Empty" << endl;
        return;
    }
    cout << "Front : " << head->data << endl;
    cout << "Rear : " << tail->data << endl;
}

void Total_Element(int size)
{
    cout << "Total Elements In Queue : " << size << endl;
}
```

```cpp
void Bars()
{
    cout << "-----------------------------------------------
----------------\n";
}
bool Options(Node *&head, Node *&tail, int &size)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Enqueue(head, tail, size);
        break;
    case 2:
        Dequeue(head, size);
        break;
    case 3:
        Front_Rear(head, tail, size);
        break;
    case 4:
        isEmpty(size);
        break;
    case 5:
        Total_Element(size);
        break;
    case 6:
        Display(head, size);
        break;
    case 7:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "_____Operations_On_Queue_____ \n";
    cout << "1.Enqueue \n";
    cout << "2.Dequeue \n";
    cout << "3.Front And Rear Element \n";
    cout << "4.isEmpty \n";
    cout << "5.Total Elements \n";
    cout << "6:Display \n";
    cout << "7.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{

    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n\n";
    int size = 0;
    Node *head = nullptr, *tail = nullptr;
    while (true)
    {
        Menu();
        if (!Options(head,tail,size))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

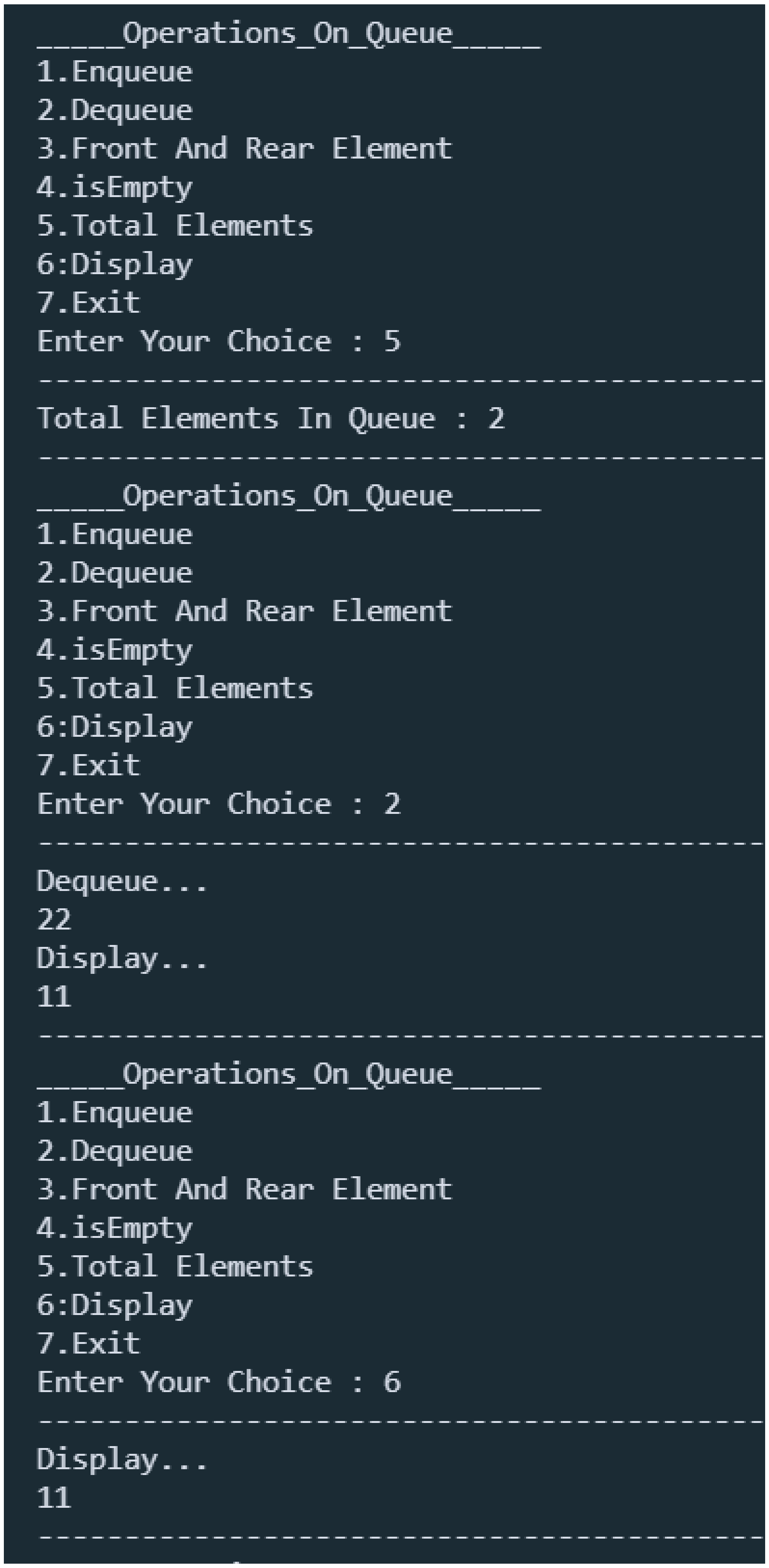
Output :-



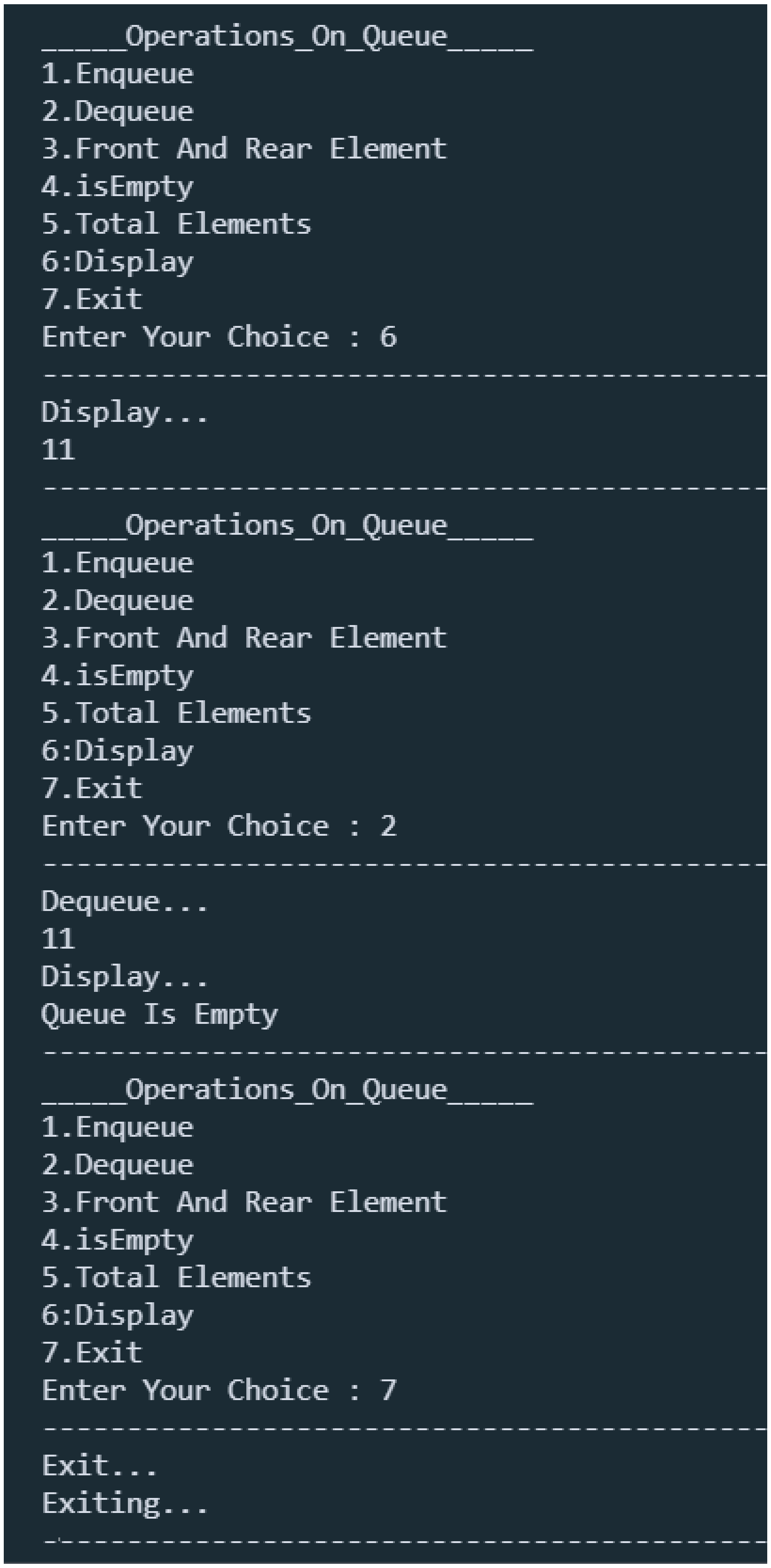
```
_____Vicky_Gupta_20BCS070_____

_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
---------------------------------------
Enqueue...
Enter The Element : 33
Display...
33
---------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
---------------------------------------
Enqueue...
Enter The Element : 22
Display...
33 22
---------------------------------------
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
----------------------------------------------
Enqueue...
Enter The Element : 11
Display...
33 22 11
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 3
----------------------------------------------
Front And Rear...
Front : 33
Rear : 11
----------------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------------
Dequeue...
33
Display...
22 11
----------------------------------------------
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 5
-----------------------------------------
Total Elements In Queue : 2
-----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
-----------------------------------------
Dequeue...
22
Display...
11
-----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 6
-----------------------------------------
Display...
11
-----------------------------------------
```



```
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 6
----------------------------------------
Display...
11
----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------
Dequeue...
11
Display...
Queue Is Empty
----------------------------------------
_____Operations_On_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 7
----------------------------------------
Exit...
Exiting...
----------------------------------------
```

Data Structure Lab

CEN-391

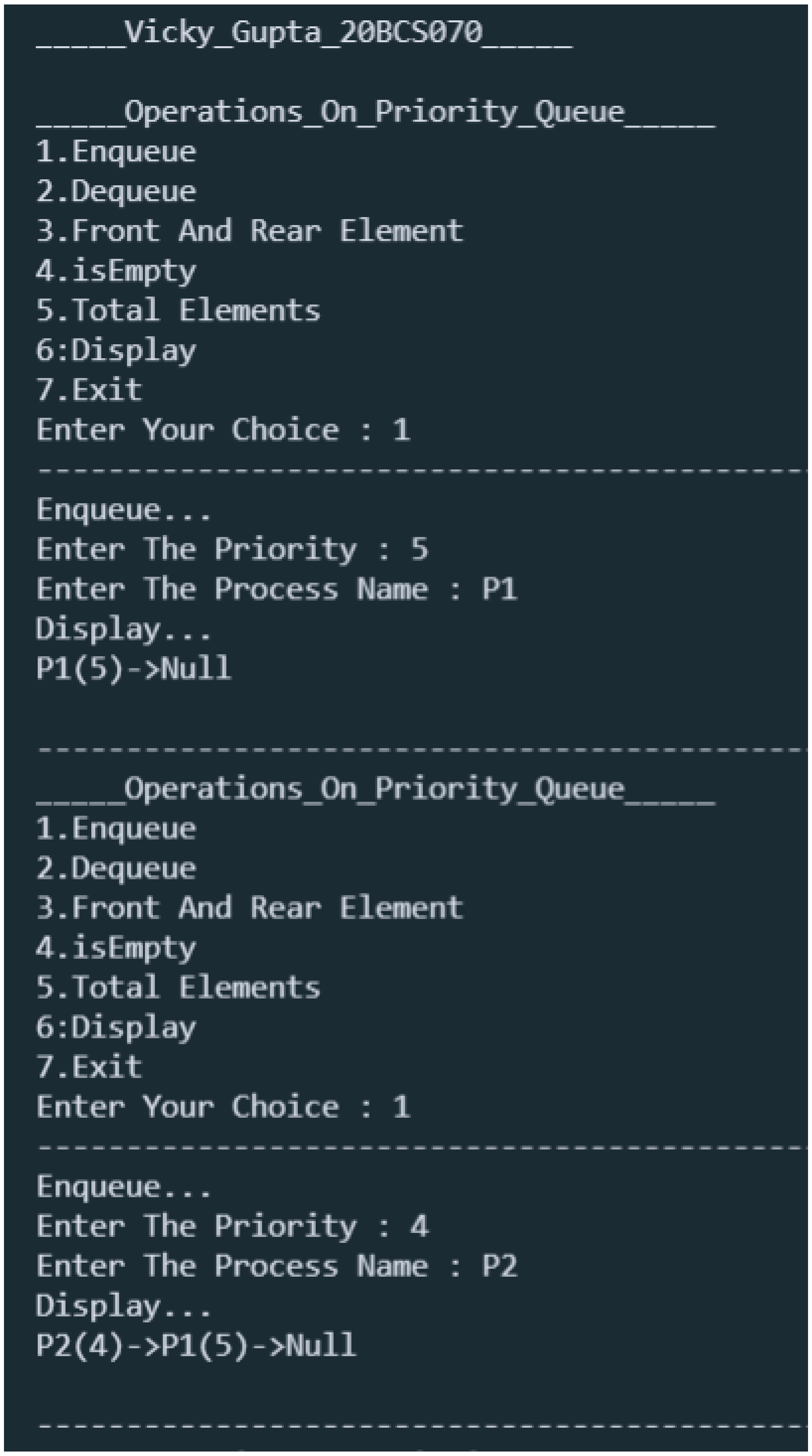
Program 10

Code :-

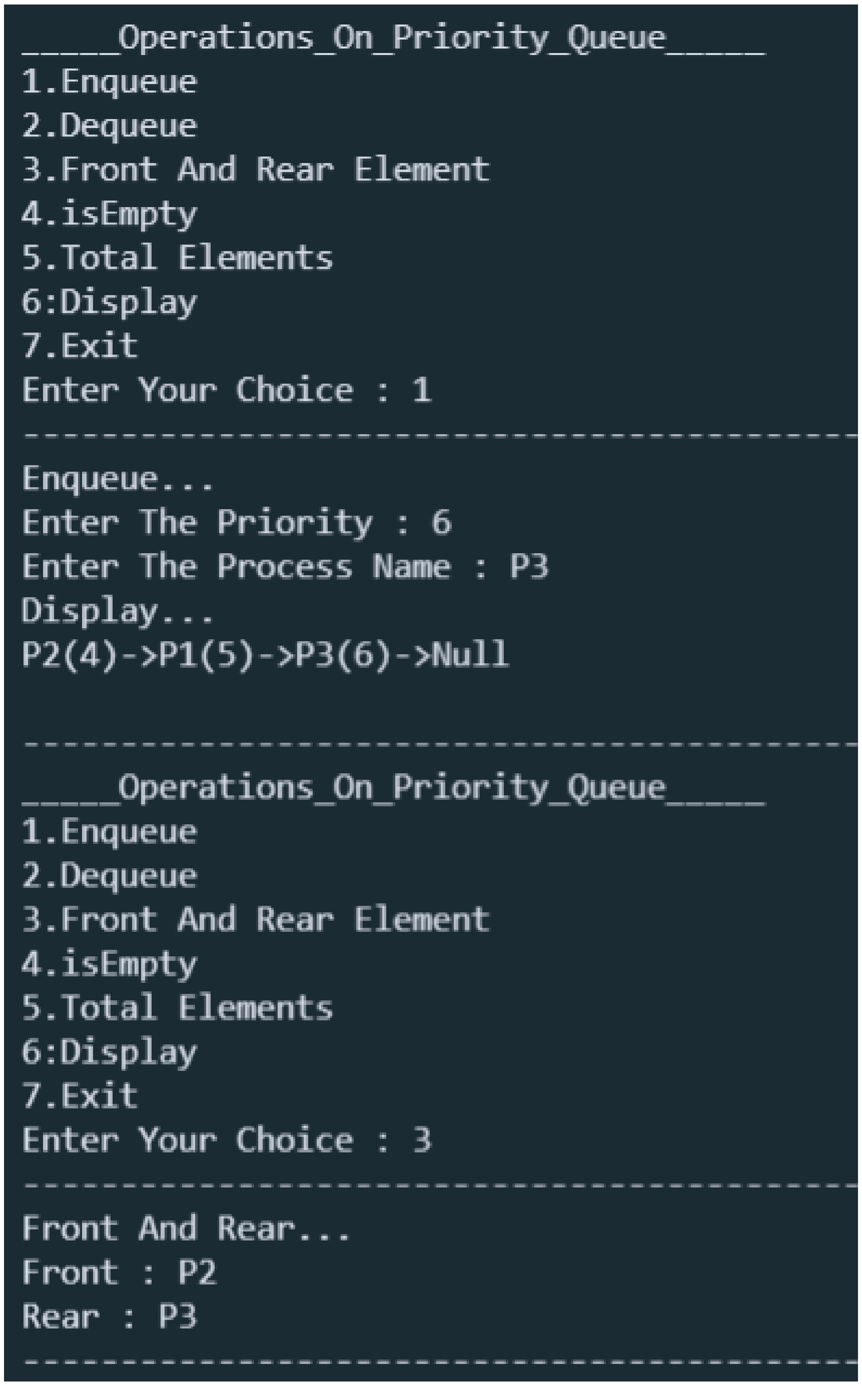
```
#include <iostream>
#include <string.h>
using namespace std;

struct Priority_Queue
{
    char process[4];
    int priority;
    Priority_Queue *next;
};

void isEmpty(int size)
{
    cout << "isEmpty...\n";
    if (size == 0)
        cout << "Empty" << endl;
    else
        cout << "Not Empty" << endl;
}

void Display(Priority_Queue *head, int size)
{
    cout << "Display...\n";
    if (size == 0)
    {
        cout << "Queue Is Empty" << endl;
        return;
    }
    while (head != nullptr)
    {
        cout << head->process << "(" << head->priority <<
")"
             << "->";
        head = head->next;
    }
    cout << "Null\n";
    cout << endl;
}

void Enqueue(Priority_Queue *&head, Priority_Queue *&tail,
int &size)
{
    cout << "Enqueue...\n";
    Priority_Queue *newnode = (Priority_Queue *)malloc(1 *
sizeof(Priority_Queue));
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned" << endl;
        return;
    }
    size++;
    cout << "Enter The Priority : ";
    int priority;
    cin >> priority;
    fflush(stdin);
    cout << "Enter The Process Name : ";
    char process[4];
    gets(process);
    strcpy(newnode->process, process);
    newnode->priority = priority;
    newnode->next = nullptr;

    Priority_Queue *temp = head;
    if (head == nullptr)
    {
        head = newnode;
        tail = newnode;
    }
    else
    {
        if (temp->priority > newnode->priority)
        {
            newnode->next = head;
            head = newnode;
        }
        else if (tail->priority <= newnode->priority)
        {
            tail->next = newnode;
            tail = tail->next;
        }
        else
        {
            while (temp && temp->next)
            {
                if (temp->next->priority > newnode-
>priority)
                {
                    newnode->next = temp->next;
                    temp->next = newnode;
                    break;
                }
                temp = temp->next;
            }
        }
    }
    Display(head, size);
}

void Dequeue(Priority_Queue *&head, int &size)
{
    cout << "Dequeue...\n";
    if (size == 0)
    {
        cout << "Queue Underflow" << endl;
        return;
    }
    cout << head->process << "(" << head->priority << ")"
         << "\n";
    size--;
    Priority_Queue *todelete = head;
    head = head->next;
    delete todelete;
    Display(head, size);
}
void Front_Rear(Priority_Queue *head, Priority_Queue *tail,
int size)
{
    cout << "Front And Rear...\n";
    if (size == 0)
    {
        cout << "Queue Is Empty" << endl;
        return;
    }
    cout << "Front : " << head->process << endl;
    cout << "Rear : " << tail->process << endl;
}

void Total_Element(int size)
{
    cout << "Total Elements In Priority Queue : " << size <<
endl;
}
```

```cpp
void Bars()
{
    cout << "-----------------------------------------------
----------------\n";
}
bool Options(Priority_Queue *&head, Priority_Queue *&tail,
int &size)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Enqueue(head, tail, size);
        break;
    case 2:
        Dequeue(head, size);
        break;
    case 3:
        Front_Rear(head, tail, size);
        break;
    case 4:
        isEmpty(size);
        break;
    case 5:
        Total_Element(size);
        break;
    case 6:
        Display(head, size);
        break;
    case 7:
        cout << "Exit...\n";
        return 0;
    default:
        cout << "Invalid Input!\nTry Again!\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "_____Operations_On_Priority_Queue_____ \n";
    cout << "1.Enqueue \n";
    cout << "2.Dequeue \n";
    cout << "3.Front And Rear Element \n";
    cout << "4.isEmpty \n";
    cout << "5.Total Elements \n";
    cout << "6:Display \n";
    cout << "7.Exit \n";
    cout << "Enter Your Choice : ";
}

int main()
{

    system("cls");
    cout << "_____Vicky_Gupta_20BCS070_____\n\n";
    int size = 0;
    Priority_Queue *head = nullptr, *tail = nullptr;
    while (true)
    {
        Menu();
        if (!Options(head, tail, size))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

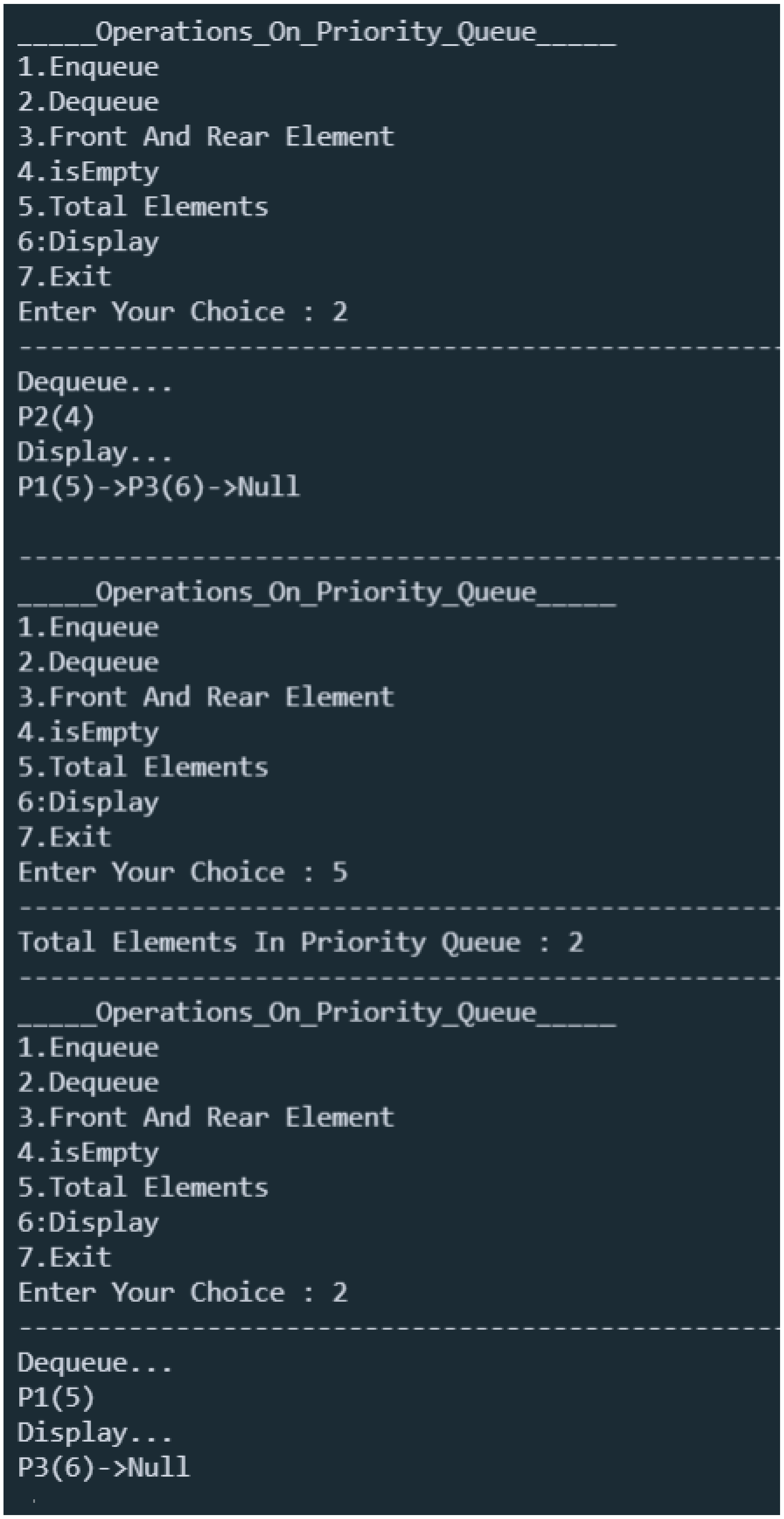
Output :-



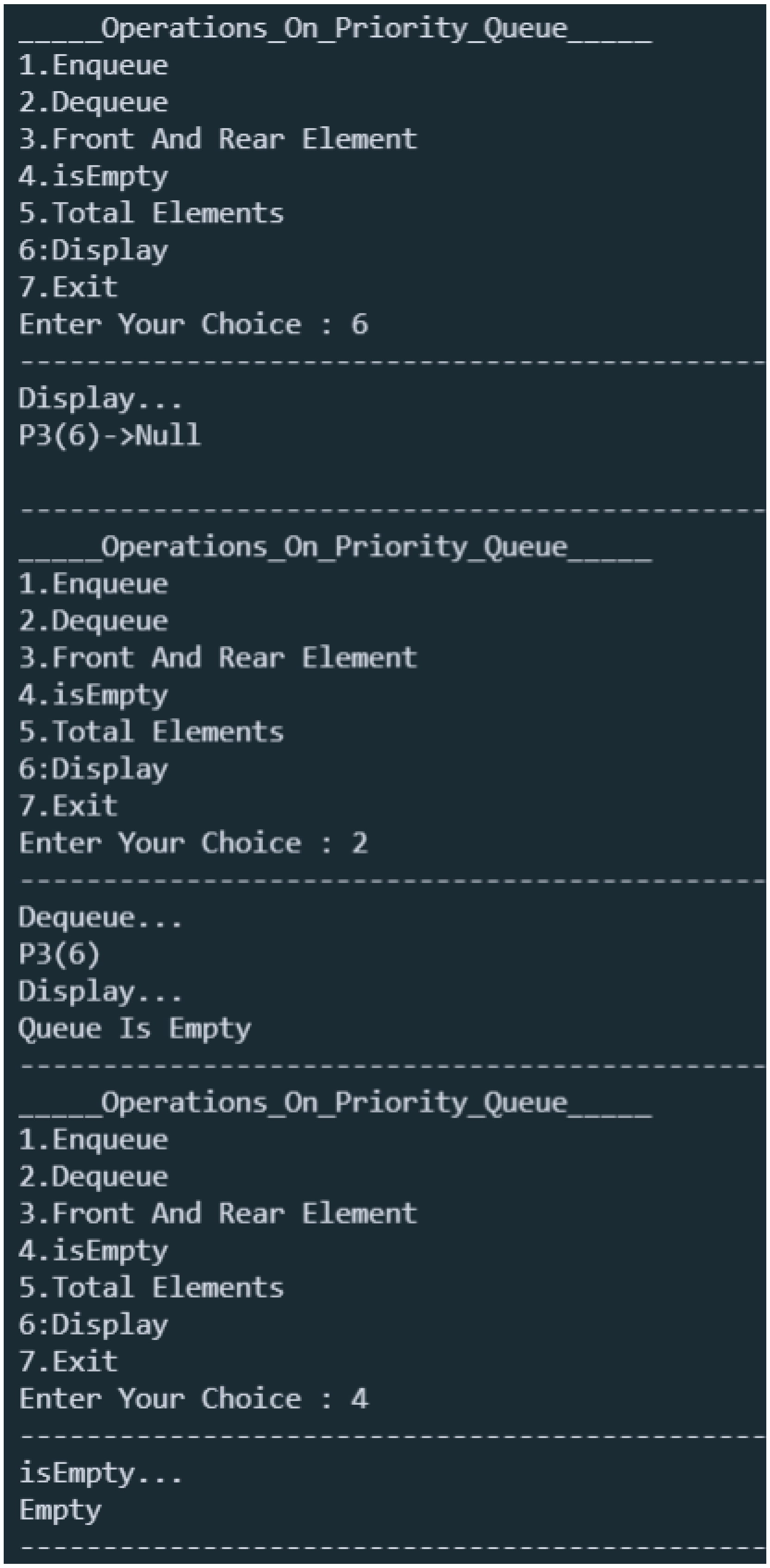
```
_____Vicky_Gupta_20BCS070_____

_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
-------------------------------------------
Enqueue...
Enter The Priority : 5
Enter The Process Name : P1
Display...
P1(5)->Null

-------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
-------------------------------------------
Enqueue...
Enter The Priority : 4
Enter The Process Name : P2
Display...
P2(4)->P1(5)->Null

-------------------------------------------
```



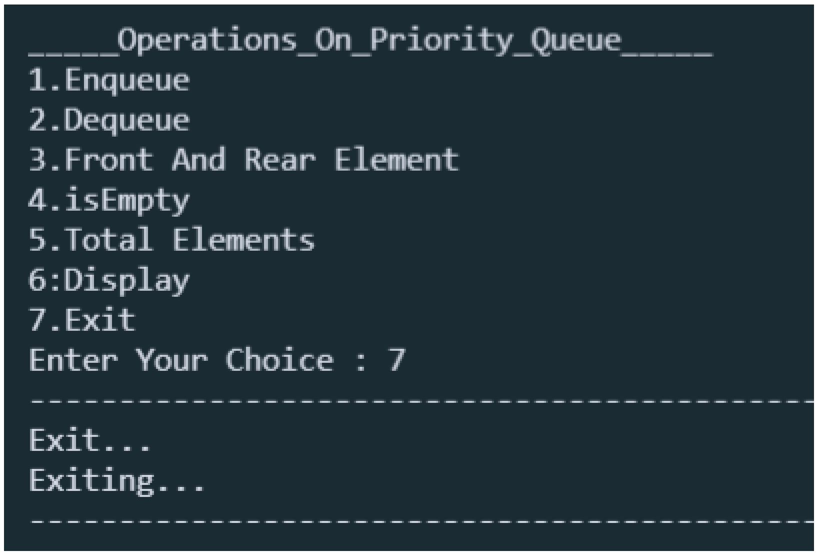
```
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 1
-------------------------------------------
Enqueue...
Enter The Priority : 6
Enter The Process Name : P3
Display...
P2(4)->P1(5)->P3(6)->Null


-------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 3
-------------------------------------------
Front And Rear...
Front : P2
Rear : P3
-------------------------------------------
```



```
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
--------------------------------------------------
Dequeue...
P2(4)
Display...
P1(5)->P3(6)->Null

--------------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 5
--------------------------------------------------
Total Elements In Priority Queue : 2
--------------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
--------------------------------------------------
Dequeue...
P1(5)
Display...
P3(6)->Null
```



```
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 6
----------------------------------------------
Display...
P3(6)->Null

----------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 2
----------------------------------------------
Dequeue...
P3(6)
Display...
Queue Is Empty
----------------------------------------------
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 4
----------------------------------------------
isEmpty...
Empty
----------------------------------------------
```



```
_____Operations_On_Priority_Queue_____
1.Enqueue
2.Dequeue
3.Front And Rear Element
4.isEmpty
5.Total Elements
6:Display
7.Exit
Enter Your Choice : 7
-------------------------------------------
Exit...
Exiting...
-------------------------------------------
```

Data Structure Lab

CEN-391

Program 11

Code :-

```
#include <iostream>
using namespace std;

struct LinkedList
{
    int data;
    LinkedList *next;
};

LinkedList *Create_NewNode()
{
    LinkedList *newnode = (LinkedList
*)malloc(sizeof(LinkedList));
    cout << "Enter The Element : ";
    cin >> newnode->data;
    newnode->next = nullptr;
```

```
        return newnode;
    }

    void Display(LinkedList *Head, int size)
    {
        cout << "Display...\n";
        if (size == 0)
        {
            cout << "Linked List Is Empty!\n";
            return;
        }
        cout << "Head";
        while (Head)
        {
            cout << "->" << Head->data << "";
            Head = Head->next;
        }
        cout << "<-Tail\n";
    }

    void Insert_At_Beginning(LinkedList *&Head, LinkedList
    *&Tail, int &size)
    {
        cout << "Insert At Beginning Operation Is Selected...
    \n";
        LinkedList *newnode = Create_NewNode();
        if (newnode == nullptr)
        {
            cout << "Memory Not Assigned!\n";
            return;
        }
        size++;
        if (Head == nullptr)
        {
            Head = newnode;
            Tail = newnode;
        }
        else
        {
```

```
        newnode->next = Head;
        Head = newnode;
    }
    Display(Head, size);
}

void Insert_At_End(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    cout << "Insert At End Operation Is Selected... \n";
    LinkedList *newnode = Create_NewNode();
    if (size == 0)
    {
        size++;
        Head = newnode;
        Tail = newnode;
        Display(Head, size);
        return;
    }
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned!\n";
        return;
    }
    size++;
    Tail->next = newnode;
    Tail = Tail->next;
    Display(Head, size);
}

void Insert_At_Given_Position(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Insert At Given Position Operation Is
Selected... \n";
    int k;
    cout << "Enter The Positon Between [0," << size << "] :
";
    cin >> k;
    if (k > size || k < 0)
    {
        cout << "Invalid Position!\n";
        return;
    }
    if (k == 0)
        Insert_At_Beginning(Head, Tail, size);
    else if (k == size)
        Insert_At_End(Head, Tail, size);
    else
    {
        size++;
        LinkedList *Current = Head, *newnode =
Create_NewNode();
        while (k > 1)
        {
            Current = Current->next;
            k--;
        }
        newnode->next = Current->next;
        Current->next = newnode;
        Display(Head, size);
    }
}

void Delete_At_Beginning(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Delete At Beginning Operation Is Selected...
\n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    size--;
    LinkedList *todelete = Head;
    Head = Head->next;
    delete todelete;
    if (size == 0)
    {
        Head == nullptr;
        Tail == nullptr;
    }
    Display(Head, size);
}

void Delete_At_End(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    cout << "Delete At End Operation Is Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    size--;
    LinkedList *Current = Head, *todelete = Tail;
    while (Current != Tail && Current->next != Tail)
    {
        Current = Current->next;
    }
    Tail = Current;
    Tail->next = nullptr;
    cout << Current->data << "\n";
    delete todelete;
    if (size == 0)
    {
        Head == nullptr;
        Tail == nullptr;
    }
    Display(Head, size);
}

void Delete_At_Given_Position(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Delete At Given Position Operation Is
Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    int k;
    cout << "Enter The Positon Between [0," << size - 1 <<
"] : ";
    cin >> k;
    if (k >= size || k < 0)
    {
        cout << "Invalid Position!\n";
        return;
    }
    if (k == 0)
        Delete_At_Beginning(Head, Tail, size);
    else if (k == size - 1)
        Delete_At_End(Head, Tail, size);
    else
    {
        size--;
        LinkedList *Current = Head, *todelete = nullptr;
        while (k > 1)
        {
            Current = Current->next;
            k--;
        }
        todelete = Current->next;
        Current->next = todelete->next;
        delete todelete;
        if (size == 0)
        {
            Head == nullptr;
            Tail == nullptr;
        }
        Display(Head, size);
    }
```

```
}

void Total_Element(int size)
{
    cout << "Total Elements Operation Is Selected... \n";
    cout << "Total Elements In Queue : " << size << endl;
}

void Search_Element(LinkedList *Head, int size)
{
    cout << "Search Element Operation Is Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Is Empty!\n";
        return;
    }
    int search;
    cout << "Enter The Element You Want To Search : ";
    cin >> search;
    int isMulti = 0;
    cout << "Do You Want To Search For Single/Multiple 
Occurence [0/1] : ";
    cin >> isMulti;
    int Position = 0;
    bool Find = false;
    while (Head)
    {
        if (Head->data == search)
        {
            Find = true;
            cout << Position << " ";
            if (isMulti == false)
                break;
        }
        Position++;
        Head = Head->next;
    }
    if (Find == false)
    {
            cout << "\nElement Not Found!\n";
        }
        else
        {
            cout << "\n"
                 << search << " Is Found At Above Positon In
Linked List\n";
        }
}

void Bars()
{
    cout << "-------------------------------------------------
----------------\n";
}

bool Options(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    int opt;
    cin >> opt;
    Bars();
    switch (opt)
    {
    case 1:
        Insert_At_Beginning(Head, Tail, size);
        break;
    case 2:
        Insert_At_End(Head, Tail, size);
        break;
    case 3:
        Insert_At_Given_Position(Head, Tail, size);
        break;
    case 4:
        Delete_At_Beginning(Head, Tail, size);
        break;
    case 5:
        Delete_At_End(Head, Tail, size);
        break;
    case 6:
        Delete_At_Given_Position(Head, Tail, size);
        break;
    case 7:
        Total_Element(size);
        break;
    case 8:
        Search_Element(Head, size);
        break;
    case 9:
        Display(Head, size);
        break;
    case 10:
        return 0;
        break;
    default:
        cout << "Invalid Input!\nTry Again!\n\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "\n_____Operations_On_Singly_Linked_List_____
\n";
    cout << "1.Insert At Beginning. \n";
    cout << "2.Insert At End. \n";
    cout << "3.Insert At Given Position. \n";
    cout << "4.Delete At Beginning. \n";
    cout << "5.Delete At End. \n";
    cout << "6.Delete At Given Position. \n";
    cout << "7.Total No Of Elements. \n";
    cout << "8.Search Of Element. \n";
    cout << "9.Display.\n";
    cout << "10.Exit.\n";
    cout << "\nEnter Your Choice : ";
}
```

```cpp
int main()
{
    system("cls");
    cout << "___Vicky_Gupta_20BCS070___\n";
    LinkedList *Head = nullptr, *Tail = nullptr;
    int size = 0;
    while (true)
    {
        Menu();
        if (!Options(Head, Tail, size))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

Output :-

```
___Vicky_Gupta_20BCS070___

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 1
----------------------------------------------
Insert At Beginning Operation Is Selected...
Enter The Element : 1
Display...
Head->1<-Tail
----------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 2
----------------------------------------------
Insert At End Operation Is Selected...
Enter The Element : 3
Display...
Head->1->3<-Tail
----------------------------------------------
```

```
_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 3
--------------------------------------------------
Insert At Given Position Operation Is Selected...
Enter The Positon Between [0,2] : 1
Enter The Element : 2
Display...
Head->1->2->3<-Tail
--------------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 7
--------------------------------------------------
Total Elements Operation Is Selected...
Total Elements In Queue : 3
--------------------------------------------------
```

```
_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 1
--------------------------------------------------------------
Insert At Beginning Operation Is Selected...
Enter The Element : 3
Display...
Head->3->1->2->3<-Tail
--------------------------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 8
--------------------------------------------------------------
Search Element Operation Is Selected...
Enter The Element You Want To Search : 3
Do You Want To Search For Single/Multiple Occurence [0/1] : 1
0 3
3 Is Found At Above Positon In Linked List
--------------------------------------------------------------
```

```
_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 6
----------------------------------------------------
Delete At Given Position Operation Is Selected...
Enter The Positon Between [0,3] : 2
Display...
Head->3->1->3<-Tail
----------------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 4
----------------------------------------------------
Delete At Beginning Operation Is Selected...
Display...
Head->1->3<-Tail
----------------------------------------------------
```

```
_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 7
-----------------------------------------------
Total Elements Operation Is Selected...
Total Elements In Queue : 2
-----------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 5
-----------------------------------------------
Delete At End Operation Is Selected...
1
Display...
Head->1<-Tail
-----------------------------------------------
```

```
_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 4
----------------------------------------------
Delete At Beginning Operation Is Selected...
Display...
Linked List Is Empty!
----------------------------------------------

_____Operations_On_Singly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Total No Of Elements.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 10
----------------------------------------------
Exiting...
----------------------------------------------
```

Data Structure Lab

CEN-391

Program 12

Code :-

```
#include <iostream>
using namespace std;

struct LinkedList
{
    int data;
    LinkedList *next;
    LinkedList *prev;
};

LinkedList *Create_NewNode()
{
    LinkedList *newnode = (LinkedList
*)malloc(sizeof(LinkedList));
    cout << "Enter The Element : ";
    cin >> newnode->data;
    newnode->next = nullptr;
    newnode->prev = nullptr;
    return newnode;
}

void Display(LinkedList *Head, int size)
{
    cout << "Display...\n";
    if (size == 0)
    {
        cout << "Linked List Is Empty!\n";
        return;
    }
    cout << "|Head|";
    while (Head)
    {
        cout << "--|" << Head->data << "|";
        Head = Head->next;
    }
    cout << "--|Tail|\n";
}

void Insert_At_Beginning(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Insert At Beginning Operation Is Selected...
\n";
    LinkedList *newnode = Create_NewNode();
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned!\n";
        return;
    }
    size++;
    if (Head == nullptr)
    {
        Head = newnode;
        Tail = newnode;
    }
    else
    {
        newnode->next = Head;
        Head->prev = newnode;
        Head = newnode;
    }
    Display(Head, size);
}

void Insert_At_End(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    cout << "Insert At End Operation Is Selected... \n";
    LinkedList *newnode = Create_NewNode();
    if (size == 0)
    {
        size++;
        Head = newnode;
        Tail = newnode;
        Display(Head, size);
        return;
    }
    if (newnode == nullptr)
    {
        cout << "Memory Not Assigned!\n";
        return;
    }
    size++;
    Tail->next = newnode;
    newnode->prev = Tail;
    Tail = Tail->next;
    Display(Head, size);
}

void Insert_At_Given_Position(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Insert At Given Position Operation Is
Selected... \n";
```

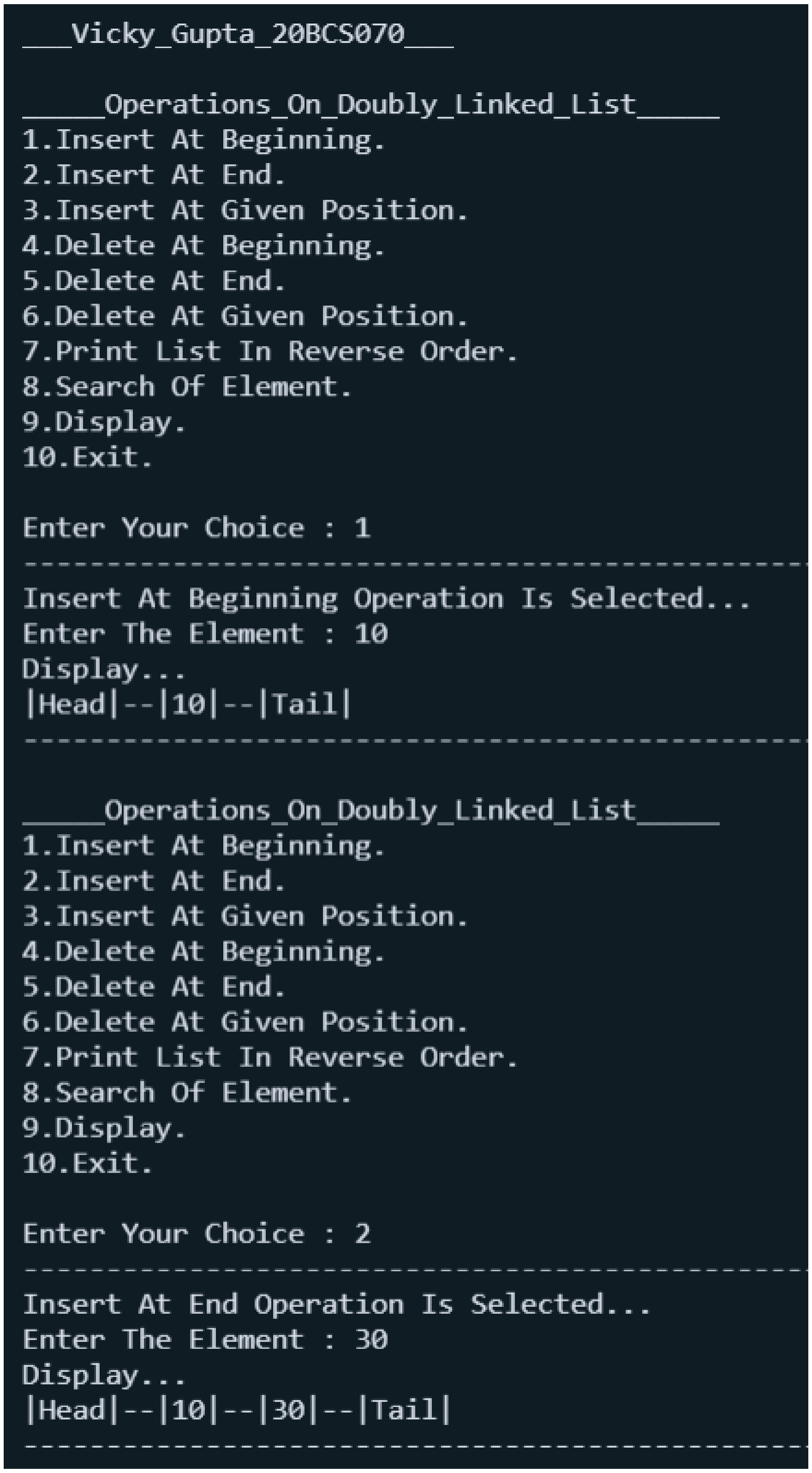
```
    int k;
    cout << "Enter The Positon Between [0," << size << "] :
";
    cin >> k;
    if (k > size || k < 0)
    {
        cout << "Invalid Position!\n";
        return;
    }
    if (k == 0)
        Insert_At_Beginning(Head, Tail, size);
    else if (k == size)
        Insert_At_End(Head, Tail, size);
    else
    {
        size++;
        LinkedList *Current = Head, *newnode =
Create_NewNode();
        while (k > 1)
        {
            Current = Current->next;
            k--;
        }
        newnode->next = Current->next;
        Current->next->prev = newnode;
        Current->next = newnode;
        newnode->prev = Current;
        Display(Head, size);
    }
}

void Delete_At_Beginning(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Delete At Beginning Operation Is Selected...
\n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    size--;
    LinkedList *todelete = Head;
    Head = Head->next;
    if (Head != nullptr)
        Head->prev = nullptr;
    delete todelete;
    if (size == 0)
    {
        Head == nullptr;
        Tail == nullptr;
    }
    Display(Head, size);
}

void Delete_At_End(LinkedList *&Head, LinkedList *&Tail, int
&size)
{
    cout << "Delete At End Operation Is Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    size--;
    LinkedList *todelete = Tail;
    Tail = Tail->prev;
    Tail->next = nullptr;
    cout << todelete->data << "\n";
    delete todelete;
    if (size == 0)
    {
        Head == nullptr;
        Tail == nullptr;
    }
    Display(Head, size);
}
```

```
void Delete_At_Given_Position(LinkedList *&Head, LinkedList
*&Tail, int &size)
{
    cout << "Delete At Given Position Operation Is
Selected... \n";
    if (size == 0)
    {
        cout << "Linked List Underflow!\n";
        return;
    }
    int k;
    cout << "Enter The Positon Between [0," << size - 1 <<
"] : ";
    cin >> k;
    if (k >= size || k < 0)
    {
        cout << "Invalid Position!\n";
        return;
    }
    if (k == 0)
        Delete_At_Beginning(Head, Tail, size);
    else if (k == size - 1)
        Delete_At_End(Head, Tail, size);
    else
    {
        size--;
        LinkedList *Current = Head, *todelete = nullptr;
        while (k > 1)
        {
            Current = Current->next;
            k--;
        }
        todelete = Current->next;
        Current->next = todelete->next;
        todelete->next->prev = Current;
        delete todelete;
        if (size == 0)
        {
            Head == nullptr;
```

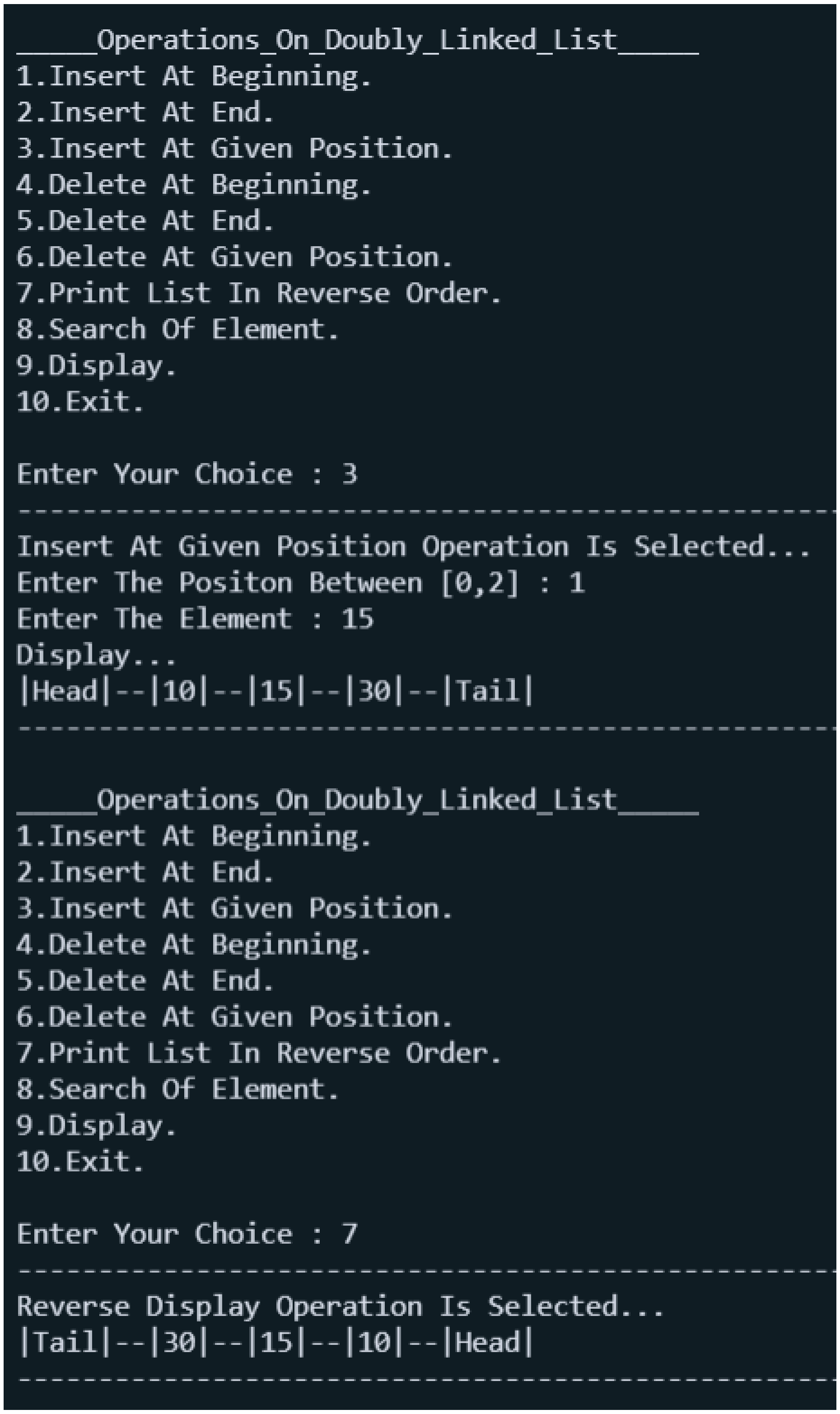
```
                Tail == nullptr;
            }
            Display(Head, size);
        }
    }

    void Reverse_Print(LinkedList *Tail, int size)
    {
        cout << "Reverse Display Operation Is Selected... \n";
        if (size == 0)
        {
            cout << "Linked List Is Empty!\n";
            return;
        }
        cout << "|Tail|";
        while (Tail)
        {
            cout << "--|" << Tail->data << "|";
            Tail = Tail->prev;
        }
        cout << "--|Head|\n";
    }

    void Search_Element(LinkedList *Head, int size)
    {
        cout << "Search Element Operation Is Selected... \n";
        if (size == 0)
        {
            cout << "Linked List Is Empty!\n";
            return;
        }
        int search;
        cout << "Enter The Element You Want To Search : ";
        cin >> search;
        int isMulti = 0;
        cout << "Do You Want To Search For Single/Multiple
    Occurence [0/1] : ";
        cin >> isMulti;
        int Position = 0;
        bool Find = false;
        while (Head)
        {
            if (Head->data == search)
            {
                Find = true;
                cout << Position << " ";
                if (isMulti == false)
                    break;
            }
            Position++;
            Head = Head->next;
        }
        if (Find == false)
        {
            cout << "\nElement Not Found!\n";
        }
        else
        {
            cout << "\n"
                 << search << " Is Found At Above Positon In 
    Linked List\n";
        }
    }

    void Bars()
    {
        cout << "-----------------------------------------------
    ----------------\n";
    }

    bool Options(LinkedList *&Head, LinkedList *&Tail, int 
    &size)
    {
        int opt;
        cin >> opt;
        Bars();
        switch (opt)
        {
```

```
    case 1:
        Insert_At_Beginning(Head, Tail, size);
        break;
    case 2:
        Insert_At_End(Head, Tail, size);
        break;
    case 3:
        Insert_At_Given_Position(Head, Tail, size);
        break;
    case 4:
        Delete_At_Beginning(Head, Tail, size);
        break;
    case 5:
        Delete_At_End(Head, Tail, size);
        break;
    case 6:
        Delete_At_Given_Position(Head, Tail, size);
        break;
    case 7:
        Reverse_Print(Tail, size);
        break;
    case 8:
        Search_Element(Head, size);
        break;
    case 9:
        Display(Head, size);
        break;
    case 10:
        return 0;
        break;
    default:
        cout << "Invalid Input!\nTry Again!\n\n";
    }
    Bars();
    return 1;
}

void Menu()
{
    cout << "\n_____Operations_On_Doubly_Linked_List_____
\n";
    cout << "1.Insert At Beginning. \n";
    cout << "2.Insert At End. \n";
    cout << "3.Insert At Given Position. \n";
    cout << "4.Delete At Beginning. \n";
    cout << "5.Delete At End. \n";
    cout << "6.Delete At Given Position. \n";
    cout << "7.Print List In Reverse Order. \n";
    cout << "8.Search Of Element. \n";
    cout << "9.Display.\n";
    cout << "10.Exit.\n";
    cout << "\nEnter Your Choice : ";
}

int main()
{
    system("cls");
    cout << "___Vicky_Gupta_20BCS070___\n";
    LinkedList *Head = nullptr, *Tail = nullptr;
    int size = 0;
    while (true)
    {
        Menu();
        if (!Options(Head, Tail, size))
            break;
    }
    cout << "Exiting...\n";
    Bars();
    return 0;
}
```

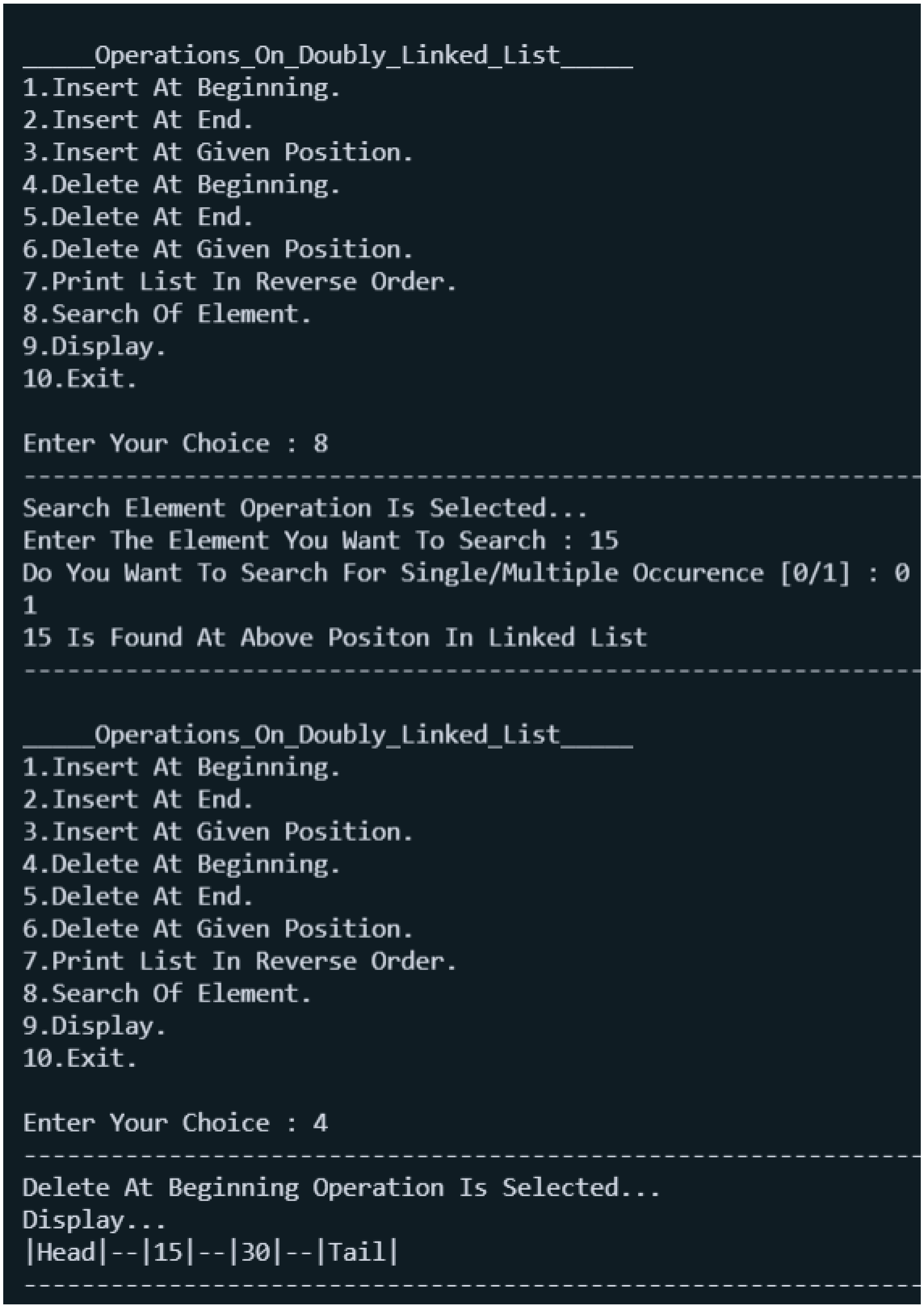
Output :-



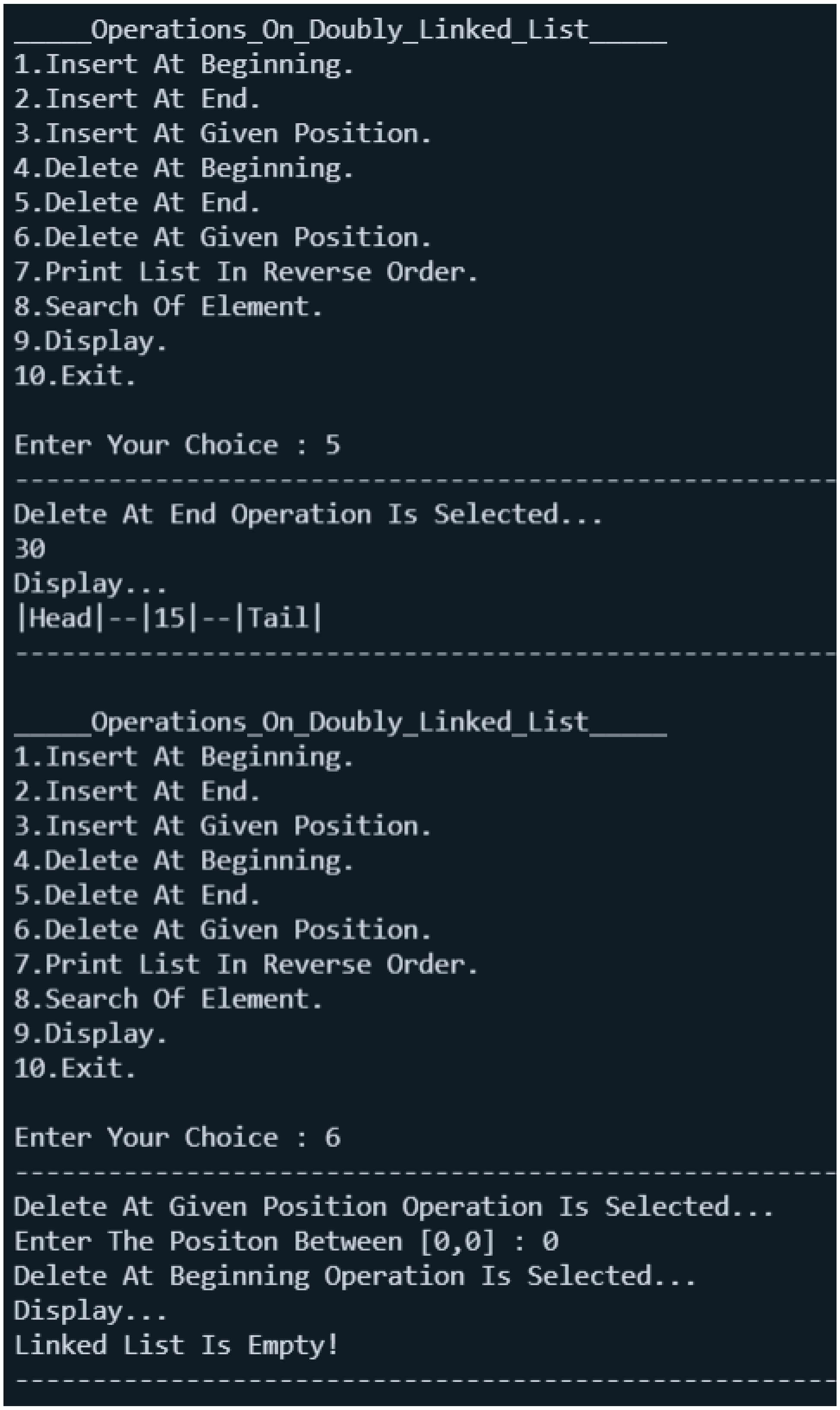
```
___Vicky_Gupta_20BCS070___

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 1
-----------------------------------------------
Insert At Beginning Operation Is Selected...
Enter The Element : 10
Display...
|Head|--|10|--|Tail|
-----------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 2
-----------------------------------------------
Insert At End Operation Is Selected...
Enter The Element : 30
Display...
|Head|--|10|--|30|--|Tail|
-----------------------------------------------
```



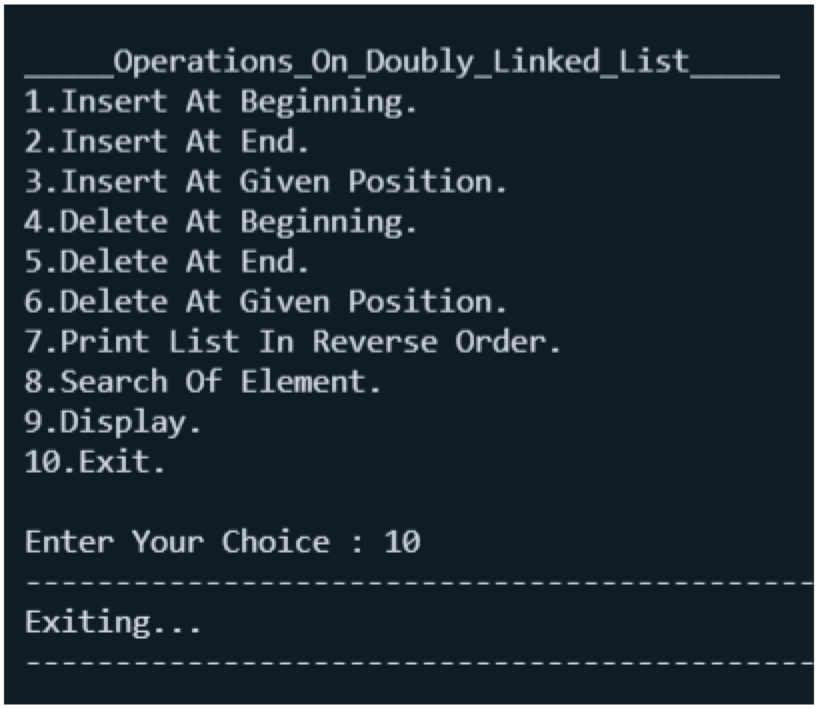
```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 3
--------------------------------------------------
Insert At Given Position Operation Is Selected...
Enter The Positon Between [0,2] : 1
Enter The Element : 15
Display...
|Head|--|10|--|15|--|30|--|Tail|
--------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 7
--------------------------------------------------
Reverse Display Operation Is Selected...
|Tail|--|30|--|15|--|10|--|Head|
--------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 8
-------------------------------------------------------------
Search Element Operation Is Selected...
Enter The Element You Want To Search : 15
Do You Want To Search For Single/Multiple Occurence [0/1] : 0
1
15 Is Found At Above Positon In Linked List
-------------------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 4
-------------------------------------------------------------
Delete At Beginning Operation Is Selected...
Display...
|Head|--|15|--|30|--|Tail|
-------------------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 5
-----------------------------------------------------
Delete At End Operation Is Selected...
30
Display...
|Head|--|15|--|Tail|
-----------------------------------------------------

_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 6
-----------------------------------------------------
Delete At Given Position Operation Is Selected...
Enter The Positon Between [0,0] : 0
Delete At Beginning Operation Is Selected...
Display...
Linked List Is Empty!
-----------------------------------------------------
```



```
_____Operations_On_Doubly_Linked_List_____
1.Insert At Beginning.
2.Insert At End.
3.Insert At Given Position.
4.Delete At Beginning.
5.Delete At End.
6.Delete At Given Position.
7.Print List In Reverse Order.
8.Search Of Element.
9.Display.
10.Exit.

Enter Your Choice : 10
---------------------------------------------
Exiting...
---------------------------------------------
```