OBJECT ORIENTED PROGRAMMING

LAB 10-Template

• Name :HITU RAJ

• Roll no. :2005025

• Branch :CSE

    /\* Q1) (Function Template) Define a function template for finding

 the minimum value contained in an array.

  Write main() function to find the minimum value of integer array

   and minimum value of floating point numbers in an array.\*/

#include <bits/stdc++.h>

using namespace std;

template <class T\_025>

void min\_025(T\_025 arr\_025[], int n\_025)

{

    T\_025 min\_element\_025 = arr\_025[0];

    for (int i\_025 = 1; i\_025 < n\_025; i\_025++)

    {

        if (arr\_025[i\_025] < min\_element\_025)

        {

            min\_element\_025 = arr\_025[i\_025];

        }

    }

    cout << "Minimum Element is: " << min\_element\_025 << endl;

}

int main()

{

    int n\_025;

    cout << "Enter the size of the Integer array: ";

    cin >> n\_025;

    int arr\_025[n\_025];

    cout << "Enter the elements of the Integer array: ";

    for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

    {

        cin >> arr\_025[i\_025];

    }

    min\_025(arr\_025, n\_025);

    cout << "Enter the size of the Floating array:";

    cin >> n\_025;

    float arr1\_025[n\_025];

    cout << "Enter the elements of the Floating array: ";

    for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

    {

        cin >> arr1\_025[i\_025];

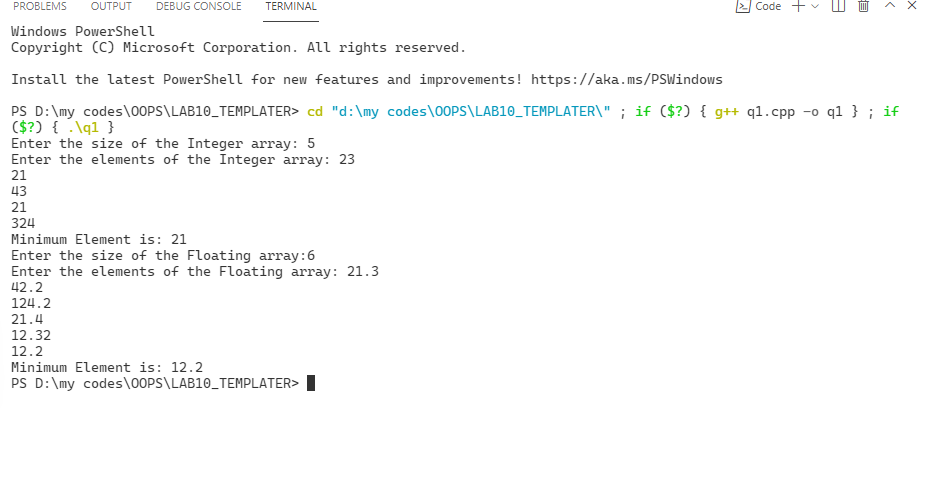
    }

    min\_025(arr1\_025, n\_025);

    return 0;

}

**OUTPUT-1**



// q2.Write a program to define the function template for

// swapping the two items of various data types such as integers, float and

// characters etc.

#include <iostream>

using namespace std;

template <class T>

void swapping(T &x, T &y)

{

    T temp = x;

    x = y;

    y = temp;

}

int main()

{

    float p, q;

    int m, n;

    char a, b;

    cout << "\n Enter two integers : ";

    cin >> m >> n;

    cout << "\nThe values of integer  before swapping are : " << m << " " << n;

    swapping(m, n);

    cout << "\nThe values of integer  after swapping are : " << m << " " << n;

    cout << ".............................................................\n";

     cout << "\n Enter two floats : ";

    cin >> p >> q;

    cout << "\nThe values before swapping are : " << p << " " << q;

    swapping(p, q);

    cout << "\nThe values after swapping are : " << p << " " << q;

    cout << ".............................................................\n";

     cout << "\n Enter two char : ";

    cin >> a >> b;

    cout << "\nThe values before swapping are : " << a << " " << b;

    swapping(a, b);

    cout << "\nThe values after swapping are : " << a << " " << b;

    return 0;

}

**OUTPUT-2**



/\*q3.Write a template function to search for a given key

element from an array. Illustrate how you perform search in integer,

character as well as double arrays using the same template function.\*/

#include <iostream>

using namespace std;

template <class T>

void search(T a[], int x, T y)

{

    int count = 0;

    for (int i = 0; i < x; i++)

    {

        if (a[i] == y)

        {

            count++;

        }

    }

    if (count > 0)

    {

        cout << "element found";

    }

    else

    {

        cout << "element not found";

    }

}

int main()

{

    float p[100], q;

    int m[100], n;

    char a[100], b;

    cout << "\n Enter the 5 value in int array : ";

    for (int i = 0; i < 5; i++)

    {

        cin >> m[i];

    }

    cout << "enter element u want to search";

    cin >> n;

    search(m, 5, n);

    cout << "........................................\n";

    cout << "\n Enter the 5 value in char array : ";

    for (int i = 0; i < 5; i++)

    {

        cin >> a[i];

    }

    cout << "enter element u want to search";

    cin >> b;

    search(a, 5, b);

    cout << "........................................\n";

    cout << "\n Enter the 5 value in flaot array : ";

    for (int i = 0; i < 5; i++)

    {

        cin >> p[i];

    }

    cout << "enter element u want to search";

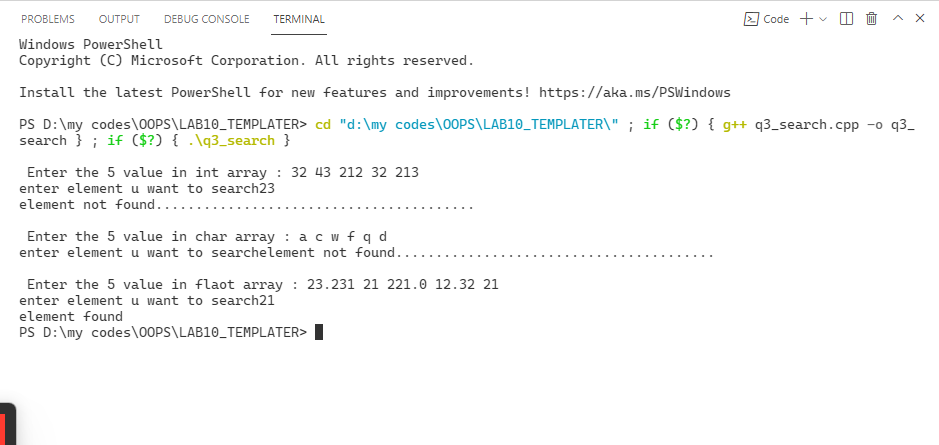
    cin >> q;

    search(p, 5, q);

    return 0;

}

**OUTPUT-3**



//Q4) (Non-type as function parameters) Write a program to demonstrate

//the concept behind function templates with non-type

//as function parameters by taking sorting an array of numbers

//as an example.

#include <bits/stdc++.h>

using namespace std;

template <class T\_025>

void sort(T\_025 \*ar\_025, int n\_025)

{

    int i\_025, j\_025;

    T\_025 temp\_025;

    for (i\_025 = 0; i\_025 < n\_025; i\_025++)

    {

        for (j\_025 = i\_025 + 1; j\_025 < n\_025; j\_025++)

        {

            if (ar\_025[i\_025] > ar\_025[j\_025])

            {

                temp\_025 = ar\_025[i\_025];

                ar\_025[i\_025] = ar\_025[j\_025];

                ar\_025[j\_025] = temp\_025;

            }

        }

    }

}

int main()

{

    int n\_025;

    cout << "Enter the size of the Array:";

    cin >> n\_025;

    int ar\_025[n\_025];

    cout << "Enter the elements of the Array: ";

    for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

    {

        cin >> ar\_025[i\_025];

    }

    sort(ar\_025, n\_025);

    cout << "Sorted Array: ";

    for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

    {

        cout << ar\_025[i\_025] << " ";

    }

    return 0;

**OUTPUT-4**



/\*Q5) (Class Template) Write a program to define a class template for reading two

data items from the keyboard and find out their sum.\*/

#include <iostream>

using namespace std;

template <class T>

class add

{

    int a, b;

public:

    T sum(T x, T y)

    {

        return x+y;

    }

};

    void SUM(int m,int n=50)

{

    m=m+n;

    n=m-n;

    cout<<m<<" "<<n;

}

int main()

{

SUM(50,10);

    /\* add<int>a;

      add<float>b;

   cout<<"sum of int is "<< a.sum(2,3)<<endl;

   cout<<"sum of float no. are "<< b.sum(2.1,3.1);\*/

    return 0;

}

**OUTPUT-5**



// Q6) (Class Template) Write a class template to represent a generic

// vector.

#include <bits/stdc++.h>

using namespace std;

template <class T\_025>

class vector\_025

{

private:

    T\_025 \*arr\_025;

    int n\_025;

public:

    vector\_025(int size\_025)

    {

        n\_025 = size\_025;

        arr\_025 = new T\_025[n\_025];

    }

    void get\_data\_025()

    {

        cout << "Enter the elements of the vector: ";

        for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

        {

            cin >> arr\_025[i\_025];

        }

    }

    void modify\_data\_025(T\_025 new\_val\_025, int index\_025)

    {

        arr\_025[index\_025] = new\_val\_025;

    }

    void multiply\_data\_025(T\_025 new\_val\_025)

    {

        for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

        {

            arr\_025[i\_025] \*= new\_val\_025;

        }

    }

    void display\_025()

    {

        cout << "The elements of the vector are: ";

        for (int i\_025 = 0; i\_025 < n\_025; i\_025++)

        {

            cout << arr\_025[i\_025] << " ";

        }

        cout << endl;

    }

};

int main()

{

    int n\_025;

    cout << "Enter the size of the vector: ";

    cin >> n\_025;

    vector\_025<int> v\_025(n\_025);

    v\_025.get\_data\_025();

    v\_025.display\_025();

    int index\_025;

    cout << "Enter the index of the element to be modified: ";

    cin >> index\_025;

    int new\_val\_025;

    cout << "Enter the new value: ";

    cin >> new\_val\_025;

    v\_025.modify\_data\_025(new\_val\_025, index\_025);

    v\_025.display\_025();

    int multiply\_val\_025;

    cout << "Enter the value to be multiplied: ";

    cin >> multiply\_val\_025;

    v\_025.multiply\_data\_025(multiply\_val\_025);

    v\_025.display\_025();

    return 0;

}

**OUTPUT-6**



/\*Q7) (Class Template) Write a program to explain class template

by creating a template T for a class named pair having two data

 members of type T which are inputted by a constructor and a member

  function get-max() return the greatest of two numbers to main.

  Note: the value of T depends upon the data type specified during

   object creation.\*/

#include <bits/stdc++.h>

using namespace std;

template <class T\_025>

class pair\_025

{

private:

    T\_025 a\_025, b\_025;

public:

    pair\_025(T\_025 x\_025, T\_025 y\_025)

    {

        a\_025 = x\_025;

        b\_025 = y\_025;

    }

    T\_025 greater\_025()

    {

        if (a\_025 > b\_025)

            return a\_025;

        else

            return b\_025;

    }

};

int main()

{

    pair\_025<int> p\_025(10, 25);

    cout << p\_025.greater\_025() << endl;

    return 0;

**OUTPUT-7**



//Q8) Matrix Implementation using Template Class. Overload Addition

//Operator.

#include <bits/stdc++.h>

 using namespace std;

template <class T\_025>

class matrix\_025

{

private:

    int r\_025, c\_025;

    T\_025 \*\*arr\_025;

public:

    matrix\_025(int row\_025, int col\_025)

    {

        r\_025 = row\_025;

        c\_025 = col\_025;

        arr\_025 = new T\_025 \*[r\_025];

        for (int i\_025 = 0; i\_025 < r\_025; i\_025++)

        {

            arr\_025[i\_025] = new T\_025[c\_025];

        }

    }

    void get\_data\_025()

    {

        cout << "Enter the elements of the matrix: \n";

        for (int i\_025 = 0; i\_025 < r\_025; i\_025++)

        {

            for (int j\_025 = 0; j\_025 < c\_025; j\_025++)

            {

                cin >> arr\_025[i\_025][j\_025];

            }

        }

    }

    matrix\_025<T\_025> operator+(matrix\_025<T\_025> m\_025)

    {

        matrix\_025<T\_025> temp\_025(r\_025, c\_025);

        for (int i\_025 = 0; i\_025 < r\_025; i\_025++)

        {

            for (int j\_025 = 0; j\_025 < c\_025; j\_025++)

            {

                temp\_025.arr\_025[i\_025][j\_025] = arr\_025[i\_025][j\_025] + m\_025.arr\_025[i\_025][j\_025];

            }

        }

        return temp\_025;

    }

    void display\_025()

    {

        cout << "The matrix is: " << endl;

        for (int i\_025 = 0; i\_025 < r\_025; i\_025++)

        {

            for (int j\_025 = 0; j\_025 < c\_025; j\_025++)

            {

                cout << arr\_025[i\_025][j\_025] << " ";

            }

            cout << endl;

        }

    }

};

int main()

{

    int row\_025, col\_025;

    cout << "Enter the number of rows and columns: ";

    cin >> row\_025 >> col\_025;

    matrix\_025<int> m1\_025(row\_025, col\_025);

    m1\_025.get\_data\_025();

    m1\_025.display\_025();

    matrix\_025<int> m2\_025(row\_025, col\_025);

    m2\_025.get\_data\_025();

    m2\_025.display\_025();

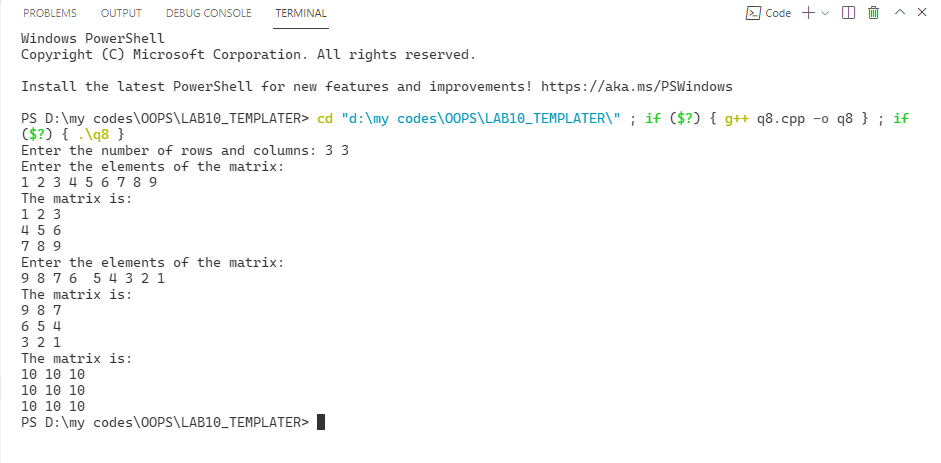
    matrix\_025<int> m3\_025 = m1\_025 + m2\_025;

    m3\_025.display\_025();

    return 0;

}

**OUTPUT-8**



//Q9) (Class Template) Design a generic stack class which can be used to

// create integer, character or floating point stack objects.

//Provide all necessary data members and member functions

//(push, pop, display &amp; default constructor) to operate on the stack.

#include <bits/stdc++.h>

 using namespace std;

template <class T\_025>

class Stack\_025

{

private:

    int top\_025;

    int capacity\_025;

    T\_025 \*array\_025;

public:

    T\_025 last\_pop\_025;

    Stack\_025(int capacity\_025)

    {

        this->capacity\_025 = capacity\_025;

        top\_025 = -1;

        array\_025 = new T\_025[capacity\_025];

        last\_pop\_025 = -9999;

    }

    void push\_025(T\_025 element\_025)

    {

        if (top\_025 == capacity\_025 - 1)

        {

            cout << "Stack Overflow" << endl;

            return;

        }

        top\_025++;

        array\_025[top\_025] = element\_025;

    }

    void pop\_025()

    {

        if (top\_025 == -1)

        {

            cout << "Stack Underflow" << endl;

            return;

        }

        int temp\_025 = top\_025--;

        last\_pop\_025 = array\_025[temp\_025];

        cout << "Popped Element - " << last\_pop\_025 << endl;

    }

    void display\_025()

    {

        if (top\_025 == -1)

        {

            cout << "Stack is Empty" << endl;

            return;

        }

        cout << "Stack Element: " << endl;

        for (int i\_025 = top\_025; i\_025 >= 0; i\_025--)

        {

            cout << array\_025[i\_025] << " ";

        }

        cout << endl;

    }

};

int main()

{

    int choice\_025, n\_025;

    cout << "Enter the size of the stack: ";

    cin >> n\_025;

    cout << "Enter 1 for Char Stack." << endl;

    cout << "Enter 2 for Integer Stack." << endl;

    cout << "Enter 3 for Float Stack." << endl;

    cout << "Enter Your Choice - ";

    cin >> choice\_025;

    switch (choice\_025)

    {

    case 1:

    {

        Stack\_025<char> s\_025(n\_025);

        char element\_025;

        while (1)

        {

            cout << "1. Push" << endl;

            cout << "2. Pop" << endl;

            cout << "3. Display" << endl;

            cout << "4. Exit" << endl;

            cout << "Enter Your Choice - ";

            cin >> choice\_025;

            switch (choice\_025)

            {

            case 1:

            {

                cout << "Enter the element to be pushed - ";

                cin >> element\_025;

                s\_025.push\_025(element\_025);

                break;

            }

            case 2:

            {

                s\_025.pop\_025();

                break;

            }

            case 3:

            {

                s\_025.display\_025();

                break;

            }

            case 4:

            {

                exit(0);

            }

            default:

            {

                cout << "Wrong Choice" << endl;

            }

            }

        }

    }

    break;

    case 2:

    {

        Stack\_025<int> s\_025(n\_025);

        int element\_025;

        while (1)

        {

            cout << "1. Push" << endl;

            cout << "2. Pop" << endl;

            cout << "3. Display" << endl;

            cout << "4. Exit" << endl;

            cout << "Enter Your Choice - ";

            cin >> choice\_025;

            switch (choice\_025)

            {

            case 1:

            {

                cout << "Enter the element to be pushed - ";

                cin >> element\_025;

                s\_025.push\_025(element\_025);

                break;

            }

            case 2:

            {

                s\_025.pop\_025();

                break;

            }

            case 3:

            {

                s\_025.display\_025();

                break;

            }

            case 4:

            {

                exit(0);

            }

            default:

            {

                cout << "Wrong Choice" << endl;

            }

            }

        }

    }

    break;

    case 3:

    {

        Stack\_025<float> s\_025(n\_025);

        float element\_025;

        while (1)

        {

            cout << "1. Push" << endl;

            cout << "2. Pop" << endl;

            cout << "3. Display" << endl;

            cout << "4. Exit" << endl;

            cout << "Enter Your Choice - ";

            cin >> choice\_025;

            switch (choice\_025)

            {

            case 1:

            {

                cout << "Enter the element to be pushed - ";

                cin >> element\_025;

                s\_025.push\_025(element\_025);

                break;

            }

            case 2:

            {

                s\_025.pop\_025();

                break;

            }

            case 3:

            {

                s\_025.display\_025();

                break;

            }

            case 4:

            {

                exit(0);

            }

            default:

            {

                cout << "Wrong Choice" << endl;

            }

            }

        }

    }

    default:

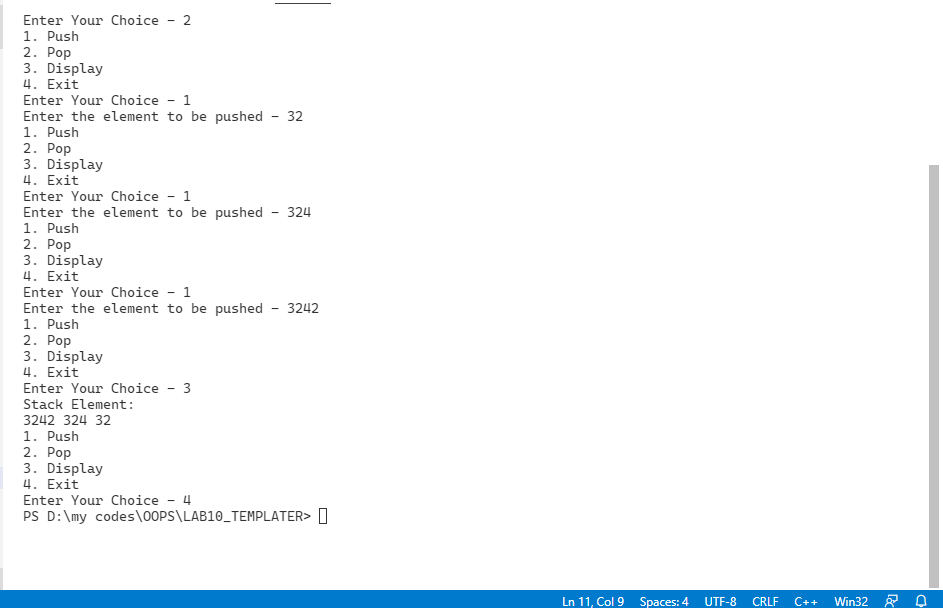
        cout << "Wrong Choice" << endl;

    }

    return 0;

}

**OUTPUT-9**



/\*Q10) (Class Template) Design a template Stack which can

 work with either a Student record or an Employee record.

  [A Student record contains name, rollNo and cgpa.

   An Employee record contains name, empId and salary fields.

    Provide push, pop, display functions to the template stack class.]\*/

#include <bits/stdc++.h>

using namespace std;

class Employee\_025

{

private:

    string name\_025;

    int id\_025;

    int salary\_025;

public:

    Employee\_025()

    {

        name\_025 = "";

        id\_025 = 0;

        salary\_025 = 0;

    }

    Employee\_025(string name\_025, int id\_025, int salary\_025)

    {

        this->name\_025 = name\_025;

        this->id\_025 = id\_025;

        this->salary\_025 = salary\_025;

    }

    void display\_025()

    {

        cout << "Name: " << name\_025 << endl;

        cout << "ID: " << id\_025 << endl;

        cout << "Salary: " << salary\_025 << endl;

    }

};

class Student\_025

{

private:

    string name\_025;

    int rollno\_025;

    int cgpa\_025;

public:

    Student\_025()

    {

        name\_025 = "";

        rollno\_025 = 0;

        cgpa\_025 = 0;

    }

    Student\_025(string name\_025, int rollno\_025, int cgpa\_025)

    {

        this->name\_025 = name\_025;

        this->rollno\_025 = rollno\_025;

        this->cgpa\_025 = cgpa\_025;

    }

    void display\_025()

    {

        cout << "Name: " << name\_025 << endl;

        cout << "Roll No: " << rollno\_025 << endl;

        cout << "CGPA: " << cgpa\_025 << endl;

    }

};

template <class T\_025>

class stack\_025

{

private:

    T\_025 \*arr\_025;

    int top\_025;

    int capacity\_025;

public:

    T\_025 last\_pop\_025;

    stack\_025(int capacity\_025)

    {

        this->capacity\_025 = capacity\_025;

        arr\_025 = new T\_025[capacity\_025];

        top\_025 = -1;

    }

    void push\_025(T\_025 ele\_025)

    {

        if (top\_025 == capacity\_025 - 1)

        {

            cout << "Stack Overflow" << endl;

            return;

        }

        arr\_025[++top\_025] = ele\_025;

    }

    void pop\_025()

    {

        if (top\_025 == -1)

        {

            cout << "Stack Underflow" << endl;

            return;

        }

        last\_pop\_025 = arr\_025[top\_025--];

    }

    void display\_025()

    {

        if (top\_025 == -1)

        {

            cout << "Stack is Empty" << endl;

            return;

        }

        for (int i\_025 = top\_025; i\_025 >= 0; i\_025--)

        {

            arr\_025[i\_025].display\_025();

        }

    }

};

int main()

{

    int choice\_025;

    while (1)

    {

        cout << "1. Employee" << endl;

        cout << "2. Student" << endl;

        cout << "3. Exit" << endl;

        cout << "Enter your choice: ";

        cin >> choice\_025;

        switch (choice\_025)

        {

        case 1:

        {

            cout << "Enter the size of the stack: ";

            int size\_025;

            cin >> size\_025;

            stack\_025<Employee\_025> s\_025(size\_025);

            choice\_025 = 0;

            while (choice\_025 != 4)

            {

                cout << "1 - Push Element to Stack.\n";

                cout << "2 - Pop Element from Stack.\n";

                cout << "3 - Display Stack.\n";

                cout << "4 - Exit.\n";

                cout << "Enter your choice: ";

                cin >> choice\_025;

                switch (choice\_025)

                {

                case 1:

                {

                    string name\_025;

                    int id\_025, salary\_025;

                    cout << "Enter the name: ";

                    cin >> name\_025;

                    cout << "Enter the id: ";

                    cin >> id\_025;

                    cout << "Enter the salary: ";

                    cin >> salary\_025;

                    s\_025.push\_025(Employee\_025(name\_025, id\_025, salary\_025));

                    break;

                }

                case 2:

                {

                    cout << "Popped Element - ";

                    s\_025.pop\_025();

                    s\_025.last\_pop\_025.display\_025();

                    break;

                }

                case 3:

                {

                    cout << "Stack : " << endl;

                    s\_025.display\_025();

                    break;

                }

                case 4:

                {

                    break;

                }

                default:

                {

                    cout << "Invalid Choice" << endl;

                }

                }

            }

        }

        break;

        case 2:

        {

            cout << "Enter the size of the stack: ";

            int size\_025;

            cin >> size\_025;

            stack\_025<Student\_025> s\_025(size\_025);

            choice\_025 = 0;

            while (choice\_025 != 4)

            {

                cout << "1 - Push Element to Stack.\n";

                cout << "2 - Pop Element from Stack.\n";

                cout << "3 - Display Stack.\n";

                cout << "4 - Exit.\n";

                cout << "Enter your choice: ";

                cin >> choice\_025;

                switch (choice\_025)

                {

                case 1:

                {

                    string name\_025;

                    int rollno\_025, cgpa\_025;

                    cout << "Enter the name: ";

                    cin >> name\_025;

                    cout << "Enter the rollno: ";

                    cin >> rollno\_025;

                    cout << "Enter the cgpa: ";

                    cin >> cgpa\_025;

                    s\_025.push\_025(Student\_025(name\_025, rollno\_025, cgpa\_025));

                    break;

                }

                case 2:

                {

                    s\_025.pop\_025();

                    break;

                }

                case 3:

                {

                    s\_025.display\_025();

                    break;

                }

                case 4:

                {

                    break;

                }

                default:

                {

                    cout << "Invalid Choice" << endl;

                }

                }

            }

        }

        break;

        case 3:

        {

            return 0;

        }

        default:

        {

            cout << "Invalid Choice" << endl;

        }

        }

    }

    return 0;

}

**OUTPUT-10**

