Date:

Assignment No.: 6

- ## **Problem Statement:**

Write a Program in C++ using class to create a stack using template.

- ## **Algorithm:**

➔ Name of the class**:** Stack,T
➔ Private data members of the class**:** count, top, T *elements
➔ Public member function of the class**:**

```
Step 1.   Stack(int):count(variable),top(0)    //Constructor
Step 2.   Void Push( T item)
Step 3.   T pop()
Step 4.   Void print()
Step 5.   ~stack()                              //Destructor
```

➔ **Algorithm for stack(int) constructor:**

```
    ✓ Set elements = new T[c]           //allocating memory
```

➔ **Algorithm for method push():**

```
Step 1.   If ( top = count )
            Then
Step 2.   Throw OVERFLOW
            [ End If ]
Step 3.   Set elements[top++] = item
```

➔ **Algorithm for method pop():**

```
Step 1.   If ( top = 0 )
            Then
Step 2.   Throw UNDERFLOW
            [ End if ]
Step 3.   Return elements[--top]
```

➔ **Algorithm for method print():**

```
•  If ( top = 0 )
            Then
•  Print "Empty"
•  Return
            [ End If ]
•  Print "elements[0]"
•  Set i=1
•  Repeat from Step 7 to Step 8 for i < top
•  Print "elements[ i ]"
•  Set i=i+1
           [ End for ]
```

➔ **Algorithm for ~Stack() destructor:**

```
➔ delete elements          // deallocates the memory provided to elements
```

➔ **Algorithm for main function():**

```
Step 1.  Print "Enter the number of elements:"
Step 2.  Read i
Step 3.  Set Stack<data_type> s = Stack<data_type> ( i )
Step 4.  Repeat from Step 5 to Step   While condition = True
Step 5.  Print "1.Push\n2.Pop\n3.Print\n4.Exit\nChoice:"
Step 6.  Read i
Step 7.  Switch( i ) do
Step 8.  Case 1:
             i.   Print "Element to push:"
             ii.  Read i
             iii. Try
                     1. s.push( i )
             iv.  catch( int i )
                     1. Print "[Error] Stack overflow!"
             v.   break
Step 9.  Case 2:
             i.   try
                     1. set j = s.pop()
                     2. Print " Element popped 'j' "
             ii.  Catch( int i )
                     1. Print "[Error] Stack underflow!"
             iii. Break
Step 10. Case 3:
    i)   Print "Elements of the stack:"
    ii)  Call s.print()
    iii) Break
Step 11. Case 4:
    i)   Return 0
    ii)  Break
Step 12. Default
    i)   Print "[ Error ] Wrong Choice"
    ii)  Break
```

• <u>Source Code:</u>

```
#include <iostream>

using namespace std;

#define MAX 100
```

```
#define OVERFLOW  0
#define UNDERFLOW 1

template <class T>
class Stack {
    public:
        Stack(int c) : count(c), top(0) {
            elements = new T[c];
        }

        void push(T item) {
            if(top == count)
                throw OVERFLOW;
            elements[top++] = item;
        }

        T pop() {
            if(top == 0)
                throw UNDERFLOW;
            return elements[--top];
        }

        void print() {
            cout << "{ ";
            if(top == 0) {
                cout << "<empty> }";
                return;
            }
            cout << elements[0];
            for(int i = 1;i < top;i++)
                cout << ", " << elements[i];
            cout << " }";
        }

        ~Stack() {
            delete elements;
        }
    private:
        int count, top;
        T *elements;
};

int main() {
    int i;
    cout << "Enter the number of elements : ";
    cin >> i;
    Stack<int> s = Stack<int>(i);
    while(1) {
        cout << "1. Push" << endl;
        cout << "2. Pop" << endl;
        cout << "3. Print" << endl;
        cout << "4. Exit" << endl;
        cout << "Choice : ";
        cin >> i;
        switch(i) {
            case 1: {
                        cout << "Element to push : ";
                        cin >> i;
                        try {
                            s.push(i);
                        } catch(int i) {
                            cout << "[Error] Stack overflow!" << endl;
                        }
                        break;
                    }
            case 2: {
                        try {
                            int j = s.pop();
                            cout << "Element popped : " << j << endl;
                        } catch(int i) {
                            cout << "[Error] Stack underflow!" << endl;
                        }
                        break;
                    }
            case 3: {
                        cout << "Elements of the stack : ";
                        s.print();
                        cout << endl;
                        break;
                    }
            case 4: {
                        return 0;
                        break;
                    }
            default: {
                        cout << "[Error] Wrong choice!" << endl;
                        break;
                     }
        }
    }
}
```

• Input & Output:

```
Enter the number of elements : 3
1. Push
2. Pop
3. Print
4. Exit
Choice : 1
```

```
Element to push : 23
1. Push
2. Pop
3. Print
4. Exit
Choice : 1
Element to push : 34
1. Push
2. Pop
3. Print
4. Exit
Choice : 1
Element to push : 40
1. Push
2. Pop
3. Print
4. Exit
Choice : 1
Element to push : 45
[Error] Stack overflow!
1. Push
2. Pop
3. Print
4. Exit
Choice : 2
Element popped : 40
1. Push
2. Pop
3. Print
4. Exit
Choice : 3
Elements of the stack : { 23, 34 }
```

- ## Discussion:

1. By the use of template our program in generalized in such a way that it can operate for any data type. We would have used function overloading but it will unnecessarily increase Line Of Code.
2. All the underflow and overflow conditions are given so there is no mismatch with the original characteristics of stack.