**ASSIGNMENT NO.:** Date:

**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**:** Stack(int)**:**count(variable),top(0) //Constructor

Void Push( T item)

T pop()

Void print()

~stack() //Destructor

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

1. Set elements = **new** T[c] //allocating memory

**Algorithm for method push():**

1. If ( top = count )

Then

1. Throw OVERFLOW

[ End If ]

1. Set elements[top++] = item

**Algorithm for method pop():**

1. If ( top = 0 )

Then

1. Throw UNDERFLOW

[ End if ]

1. Return elements[--top]

**Algorithm for method print():**

1. If ( top = 0 )

Then

1. Print “Empty”
2. Return

[ End If ]

1. Print “elements[0]”
2. Set i=1
3. Repeat from Step 7 to Step 8 for i < top
4. Print “elements[ i ]”
5. Set i=i+1

[ End for ]

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

1. **delete** elements // deallocates the memory provided to elements

**Algorithm for main function():**

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

**SOURCE CODE:**

#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 : 4

1. Push

2. Pop

3. Print

4. Exit

Choice : 1

Element to push : 10

1. Push

2. Pop

3. Print

4. Exit

Choice : 1

Element to push : 20

1. Push

2. Pop

3. Print

4. Exit

Choice : 1

Element to push : 30

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 : 50

[Error] Stack overflow!

1. Push

2. Pop

3. Print

4. Exit

Choice : 3

Elements of the stack : { 10, 20, 30, 40 }

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 : { 10, 20, 30 }

1. Push

2. Pop

3. Print

4. Exit

Choice : 4

**DISCUSSION:**

* 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 LOC.