Lab No.10 Templates and Exceptions

10.1 Introduction

Template is usually used to represent a pattern. It happens in programming that group of functions and/or classes differ from each other in terms of parameter types only. Template can be used in such situations to avoid redundant coding. This lab covers function and class template. Exceptions are runtime errors that change the regular flow of program execution. The response to these errors is known as exception handling. This lab also covers exception handling in detail.

10.2 Objectives of the lab:

1 Develop generic function using function template.

2 Develop generic class using class template.

3 Understand various types of exceptions and exception handling mechanism.

4 Use of try-catch blocks to handle exceptions.

10.3 Pre-Lab

10.3.1 Function Template

The syntax of generic function is:

```
template <class T> or template <typename T>
T function_name(T t){
    return t*t;
}
```

10.3.2 Class Template

The syntax of generic class is:

```
template <class T> or template <typename T>
class Example{
    private:
            T var;
    public:
            Example(T var){
                    this.var = var;
            }
}
```

10.3.3 Example: tempExp.cpp

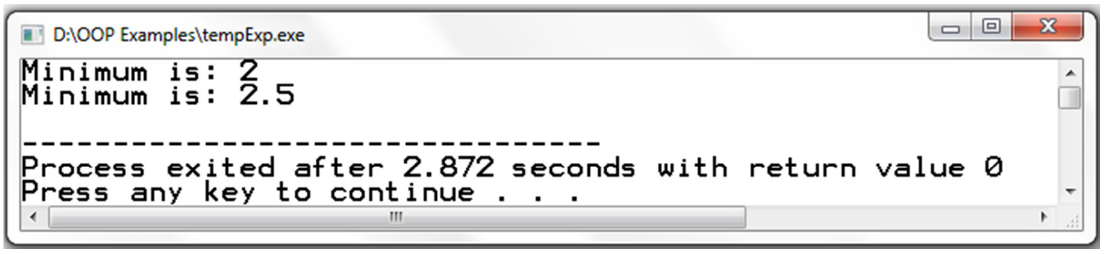
```
#include<iostream>
using namespace std;

template<class T>
class minmax{
    private:
            T n1, n2;
    public:
            minmax(T one, T two){
                    n1 = one;
                    n2 = two;
            }
            T getmin();
};

template<class T>
T minmax<T>::getmin(){
    T res;
    if(n1<n2){
            res = n1;
    }
    else{
            res = n2;
    }
    return res;
}

int main(){
    minmax<int> m1(2,3);
    cout << "Minimum is: " << m1.getmin() << endl;
    minmax<double> m2(2.7,2.5);
    cout << "Minimum is: " << m2.getmin() << endl;
    return 0;
}
```

Output:



```
Minimum is: 2
Minimum is: 2.5

--------------------------------
Process exited after 2.872 seconds with return value 0
Press any key to continue . . .
```

Figure 10.1: Output tempExp.cpp

10.3.4 Exception Handling

The syntax of exception handling is:

```
try(){
    // code statements here
} catch(type e){
    // code to handle the exception
}
```

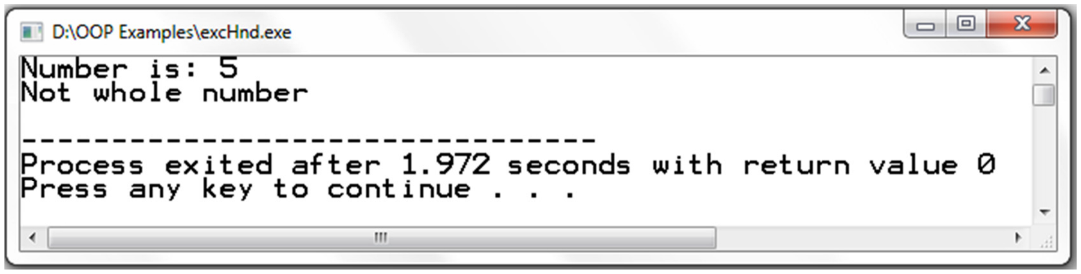
10.3.5 Example: excHnd.cpp

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

class WholeNum{
    private:
        int num;
    public:
        WholeNum(int n){
            num = n;
        }
        void checkNum(){
            if(num<0){
                throw "Not whole number";
            }
            else{
                cout << "Number is: " << num << endl;
            }
        }
};

int main(){
    WholeNum w1(5);
    WholeNum w2(-5);
    try{
        w1.checkNum();
        w2.checkNum();
    }
    catch(const char* msg){
        cerr << msg << endl;
    }
    return 0;
}
```

Output:



```
Number is: 5
Not whole number

--------------------------------
Process exited after 1.972 seconds with return value 0
Press any key to continue . . .
```

Figure 10.1: Output excHnd.cpp

10.4 Activities

Perform these activities in C++ only.

Bonus marks for those who submit in C++, JAVA, and Python.

10.4.1 Activity

Create a function called **Swap** that interchanges the values of the two arguments sent to it. (You will probably want to pass these arguments by reference.) Make the function into a template, so it can be used with all numerical data types (char, int, float, and so on). Write a main() program to exercise the function with several types.

10.4.2 Activity

Create a class called **SmartArray** that can store data of any type and generates exception when it overflows. The class should contain the following stuff:

1. Two data members, A pointer for array and an integer variable to store the size of array.
2. No argument constructor that initializes size to some default value and allocate memory for that pointer according to that size.
3. One argument constructor that initializes size to the value passed and allocate memory for that pointer according to that passed size.
4. A member function called access that give direct access to the array by returning a reference.
5. The function should throw exceptions if the index passed to the access function is out of bound of the array.
6. A destructor that deletes the allocated memory.

Write a test program in main that tests all the features of the class.

10.4.3 Activity

Create a class Rectangle. The class has attributes length and width, each of which defaults to 1. Provide methods that calculate the perimeter and the area of the rectangle. Provide set and get methods for both length and width. The set methods should verify that length and width are each floating-point numbers greater than or equal to 0.0 and less than 20.0. Write a program to test class Rectangle.

Next, add an exception class Range so that out of range initialized values of length and width will trigger the exception. The catch block for an exception prints an error message for the user.

10.5 Testing

Test Cases for Activity 10.4.1 to 10.4.3

Sample Inputs	Sample Outputs
Test for inputs of your choice and show results.	

10.6 References:

1. **Class notes**
2. **Object-Oriented Programming in C++ by *Robert Lafore***
3. **How to Program C++ by *Deitel & Deitel***
4. **Programming and Problem Solving with Java by *Nell Dale & Chip Weems***
5. **Murach's Python Programming by *Micheal Urban & Joel Murach***