Lab Exercise - 2

The program template give some tips about the program structure. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, write your C++ code. Compile and execute the program. Compare your output with the sample output provided.

Lab Objectives

In this lab, you will practice:
• Creating member functions.
• Invoking functions and receiving return values from functions.
• Testing a condition using an if statement.
• Outputting variables with stream insertion and the cout object

Description of the Problem

Write a random number generator class that generates random numbers depending on the called functions and input arguments. Your class header file is given to you. Your job is to implement these class member functions and obtain required outputs. You can compare your output by the given sample output. Some of the test procedures is given to you (main.cpp). You are also required to complete missing test functions.

UML Diagram

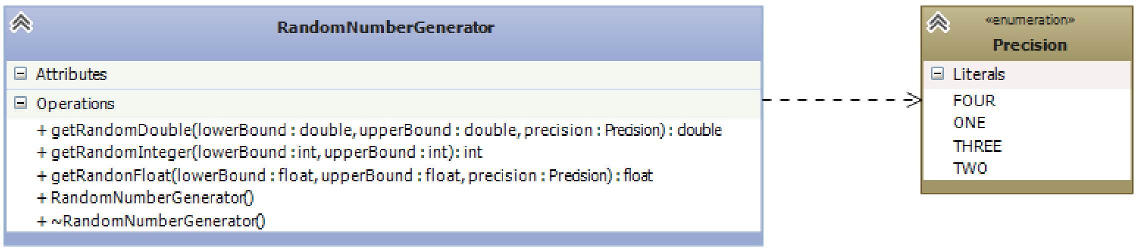
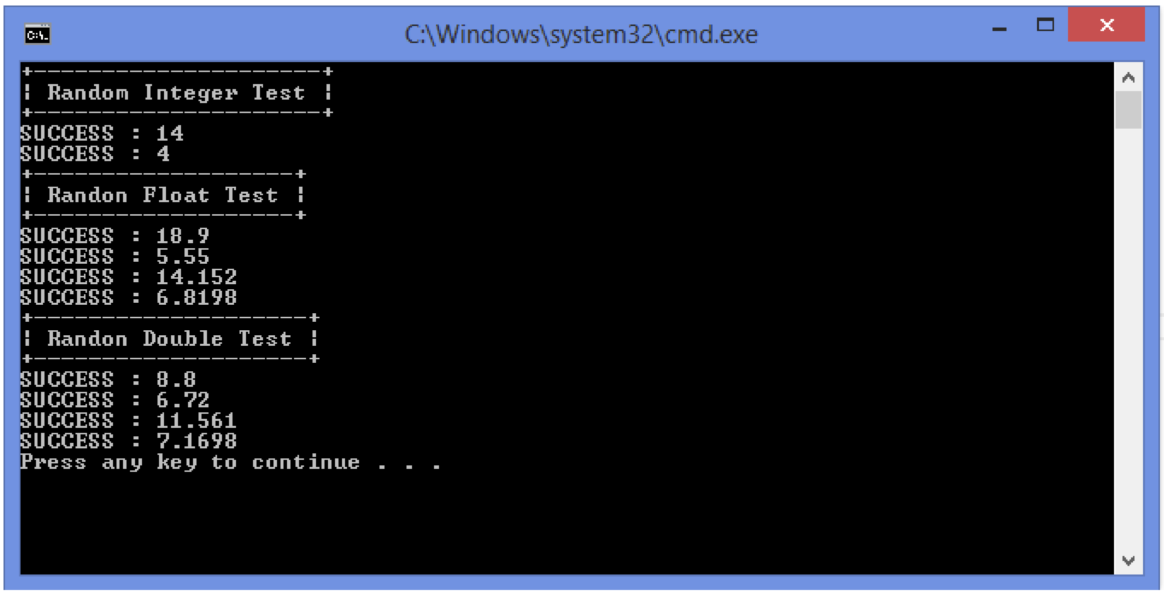

RandomNumberGenerator
Attributes
Operations
+ getRandomDouble(lowerBound : double, upperBound : double, precision : Precision) : double
+ getRandomInteger(lowerBound : int, upperBound : int): int
+ getRandonFloat(lowerBound : float, upperBound : float, precision : Precision) : float
+ RandomNumberGenerator()
+ ~RandomNumberGenerator()
«enumeration»
Precision
Literals
FOUR
ONE
THREE
TWO

Figure 1 UML Diagram

Sample Output



```
+------------------------+
| Random Integer Test |
+------------------------+
SUCCESS : 14
SUCCESS : 4
+----------------------+
| Randon Float Test |
+----------------------+
SUCCESS : 18.9
SUCCESS : 5.55
SUCCESS : 14.152
SUCCESS : 6.8198
+-----------------------+
| Randon Double Test |
+-----------------------+
SUCCESS : 8.8
SUCCESS : 6.72
SUCCESS : 11.561
SUCCESS : 7.1698
Press any key to continue . . .
```

Figure 2 Sample Output

Template

```
/// RandomNumberGenerator.h
#pragma once
class RandomNumberGenerator
{
public:
	/// <summary>
	/// Precision of a float or double number
	/// ONE : one digit after point
	/// TWO : two digit after point
	/// THREE : three digit after point
	/// FOUT : four digit after point
	/// </summary>
	enum Precision
	{
		/// <summary>
		/// The one
		/// </summary>
		ONE,
		/// <summary>
		/// The two
		/// </summary>
		TWO,

		/// <summary>
		/// The three
		/// </summary>
		THREE,
		/// <summary>
		/// The four
		/// </summary>
		FOUR
	};

	/// <summary>
	/// Initializes a new instance of the <see cref="RandomNumberGenerator"/> class.
	/// </summary>
	RandomNumberGenerator();
	/// <summary>
	/// Finalizes an instance of the <see cref="RandomNumberGenerator"/> class.
	/// </summary>
	virtual ~RandomNumberGenerator();
	/// <summary>
	/// Gets the random integer.
	/// </summary>
	/// <param name="lowerBound">The lower bound.</param>
	/// <param name="upperBound">The upper bound.</param>
	/// <returns></returns>
	int getRandomInteger(int lowerBound, int upperBound);
	/// <summary>
	/// Gets the random float.
	/// </summary>
	/// <param name="lowerBound">The lower bound.</param>
	/// <param name="upperBound">The upper bound.</param>
	/// <param name="precision">The precision.</param>
	/// <returns></returns>
	float getRandomFloat(float lowerBound, float upperBound, Precision precision);
	/// <summary>
	/// Gets the random double.
	/// </summary>
	/// <param name="lowerBound">The lower bound.</param>
	/// <param name="upperBound">The upper bound.</param>
	/// <param name="precision">The precision.</param>
	/// <returns></returns>
	double getRandomDouble(double lowerBound, double upperBound, Precision
	precision);
};
```

```
/// RandomNumberGenerator.cpp
#include "RandomNumberGenerator.h"
#include <time.h>
#include <iostream>
using namespace std;

RandomNumberGenerator::RandomNumberGenerator()
{
srand(time(NULL));
}
RandomNumberGenerator::~RandomNumberGenerator()
{
}
int RandomNumberGenerator::getRandomInteger(int lowerBound, int upperBound)
{
	// Implement the function
}
float RandomNumberGenerator::getRandomFloat(float lowerBound, float upperBound,
Precision precision)
{
	// Implement the function
	// You are required to use switch case structure for the enum type
}
double RandomNumberGenerator::getRandomDouble(double lowerBound, double upperBound,
Precision precision)
{
	// Implement the function
	// You are required to use switch case structure for the enum type
}
```

```
/// RandomNumberGeneratorTestMain.cpp
#include <iostream>
#include "RandomNumberGenerator.h"
using namespace std;
/// <summary>
/// Random Integer Test
/// </summary>
/// <param name="generator">The generator.</param>
/// <param name="lowerBound">The lower bound.</param>
/// <param name="upperBound">The upper bound.</param>
void TEST_RandomInteger(RandomNumberGenerator& generator, int lowerBound, int
upperBound)
{

	int randomNumber = generator.getRandomInteger(lowerBound, upperBound);
	if (randomNumber >= lowerBound && randomNumber <= upperBound)
	{
        cout << "SUCCESS : " << randomNumber << endl;
        }
        else
        {

        cout << "FAILURE : Obtained number is not between the range [" << lowerBound << "," << upperBound <<
        "]" << endl;

        }

}



/// <summary>
/// Random Float Test
/// </summary>
/// <param name="generator">The generator.</param>
/// <param name="lowerBound">The lower bound.</param>
/// <param name="upperBound">The upper bound.</param>
/// <param name="precision">The precision.</param>
void TEST_RandomFloat(RandomNumberGenerator& generator, float lowerBound, float
upperBound, RandomNumberGenerator::Precision precision)
{
// Implement the function
}

/// <summary>
/// Random Double Test
/// </summary>
/// <param name="generator">The generator.</param>
/// <param name="lowerBound">The lower bound.</param>
/// <param name="upperBound">The upper bound.</param>
/// <param name="precision">The precision.</param>
void TEST_RandomDouble(RandomNumberGenerator& generator, double lowerBound, double
upperBound, RandomNumberGenerator::Precision precision)
{
// Implement the function
}
/// <summary>
/// Main Function
/// </summary>
/// <returns></returns>
int main()
{

        RandomNumberGenerator generator;
        cout << "+---------------------+" << endl
        << "| Random Integer Test |" << endl
        << "+---------------------+" << endl;
        TEST_RandomInteger(generator,5,20);
        TEST_RandomInteger(generator, 2, 60);
        cout <<"+-------------------+" << endl
        << "| Randon Float Test |" << endl
        << "+-------------------+" << endl;
        TEST_RandomFloat(generator, 5, 20, RandomNumberGenerator::Precision::ONE);
        TEST_RandomFloat(generator, 5, 20, RandomNumberGenerator::Precision::TWO);
        TEST_RandomFloat(generator, 5, 20, RandomNumberGenerator::Precision::THREE);
        TEST_RandomFloat(generator, 5, 20, RandomNumberGenerator::Precision::FOUR);

        cout << "+--------------------+" << endl
        << "| Randon Double Test |" << endl
        << "+--------------------+" << endl;
        TEST_RandomDouble(generator, 5, 20, RandomNumberGenerator::Precision::ONE);
```

```
    TEST_RandomDouble(generator, 5, 20, RandomNumberGenerator::Precision::TWO);
    TEST_RandomDouble(generator, 5, 20, RandomNumberGenerator::Precision::THREE);
    TEST_RandomDouble(generator, 5, 20, RandomNumberGenerator::Precision::FOUR);

}
```

Problem-Solving Tips

1- To obtain a number between the given range. You can check the previous lab tips.

2- To obtain a float number with a given precision. You can apply the following algorith.

Assume : Precision value is equel to “ONE”. That is, these will be one digit after decimal point.

- Multiply both lower and upper bound variables by “10” and store them different variables of type integer.
- Obtain a random integer value using multiplied boundaries and store it in a float variable
- Divide obtained float value by “10”.
- Return value.

Same Algorith can be applied to double number generation.

3- To implement test function, use given test function sample.