Experiment 3
Classes and Objects - I

Objectives

To write simple computer programs using classes and objects.

Prelab Activities

Programming Output

For each of the given program segments, read the code and write the output in the space provided below each program. [Note: Do not execute these programs on a computer.]

For Programming Output Exercises 1–5, use the following class definition.

```
1 // Definition of Account class.
2 class Account
3 {
4 public:
5     Account( int ); // constructor initializes balance
6     void credit( int ); // add an amount to the account balance
7     int getBalance(); // return the account balance
8  private:
9     int balance; // data member that stores the balance
10 }; // end class Account
```

```
1  // Member-function definitions for class Account.
2  #include <iostream>
3  using namespace std;
4
5  #include "Account.h" // include definition of class Account
6
7  // Account constructor initializes data member balance
8  Account::Account( int initialBalance )
9  {
10           balance = 0; // assume that the balance begins at 0
11
12           // if initialBalance is greater than 0, set this value as the
13           // balance of the Account; otherwise, balance remains 0
14           if ( initialBalance > 0 )
15                 balance = initialBalance;
16
17           // if initialBalance is negative, print error message
18           if ( initialBalance < 0 )
19                 cout << "Error: Initial balance cannot be negative.\n" << endl;
20    } // end Account constructor
21
22    // credit (add) an amount to the account balance
23    void Account::credit( int amount )
24    {
25          balance = balance + amount; // add amount to balance
26    } // end function credit
27
28    // return the account balance
29    int Account::getBalance()
30    {
31          return balance; // gives the value of balance to the calling function
32 } // end function getBalance
```

1. What is output by the following main function?

```
1 int main()
2 {
3     Account account1( 3550 );
4
5     cout << "account1 balance: $" << account1.getBalance()  << endl;
6 } // end main
```

2. What is output by the following main function?

```
1 int main()
2 {
3     Account account1( -2017 );
4
5     cout << "account1 balance: $" << account1.getBalance()  << endl;
6 } // end main
```

3. What is output by the following main function?

```
1 int main()
2 {
3     Account account1( 1533 );
4
5     cout << "account1 balance: $" << account1.getBalance()  << endl;
6     cout << "adding $253 to account1 balance" << endl;
7
8     account1.credit( 253 );
9     cout << "account1 balance: $" << account1.getBalance()  << endl;
10 } // end main
```

4. What is output by the following main function?

```
1 int main()
2 {
3     Account account1( 2770 );
4
5     cout << "account1 balance: $" << account1.getBalance()  << endl;
6     cout << "adding $375 to account1 balance" << endl;
7
8     account1.credit( 375 );
9     cout << "account1 balance: $" << account1.getBalance()  << endl;
10 } // end main
```

5. What is output by the following main function?

```
1 int main()
2 {
3     Account account1( 799 );
4
5     cout << "account1 balance: $" << account1.getBalance()  << endl;
6     cout << "adding -$114 to account1 balance" << endl;
7
8     account1.credit( -114 );
9     cout << "account1 balance: $" << account1.getBalance()  << endl;
10 } // end main
```

CorrectheCode

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, write the corrected code. If the code does not contain an error, write "no error." For code segments, assume the code appears in main and that using directives are provided. [*Note*: It is possible that a program segment may contain multiple errors.]

For Correct the Code Exercises 6–9, use the following class definition.

```
1  // Definition of GradeBook class that stores the course name.
2  #include <string> // program uses C++ standard string class
3  using namespace std;
4
5  // GradeBook class definition
6  class GradeBook
7  {
8  public:
9     // constructor initializes course name and instructor name
10      GradeBook( string, string );
11    void setCourseName( string ); // function to set the course name
12    string getCourseName(); // function to retrieve the course name
13    void displayMessage(); // display welcome message and course name
14 private:
15    string courseName; // course name for this GradeBook
16 }; // end class GradeBook
```

```
1  // Member-function definitions for class GradeBook.
2  #include <iostream>
3  using namespace std;
4
5  // include definition of class GradeBook from GradeBook.h
6  #include "GradeBook.h"
7
8  // constructor initializes courseName
9  // with string supplied as argument
10 GradeBook::GradeBook( string course )
11 {
12    setCourseName( course ); // initializes courseName
13 } // end GradeBook constructor
14
15 // function to set the course name
16 void GradeBook::setCourseName( string name )
17 {
18    courseName = name; // store the course name
19 } // end function setCourseName
20
21 // function to retrieve the course name
22 string GradeBook::getCourseName()
23 {
24    return courseName;
25 } // end function getCourseName
26
27 // display a welcome message and the course name
28 void GradeBook::displayMessage()
29 {
30    // display a welcome message containing the course name
31    cout << "Welcome to the grade book for\n" << getCourseName() << "!"
32    << endl;
33 } // end function displayMessage
```

6. The following code segment should create a new GradeBook object:

```
Gradebook gradeBook( "Introduction to C++", 25 );
```

7. The following code segment should set the course name for the gradeBook object:

```
setCourseName( gradeBook, "Advanced C++" )
```

8. The following code segment should ask the user to input a course name. That name should then be set as the course name of your gradeBook.

```
1 cout << "Please enter the course name:" << endl;
2 cin.getline( inputName );
3
4 gradeBook.setCourseName();
```

9. The following code segment should output gradeBook's current course name:

```
cout << "The grade book's course name is: " <<gradeBook.courseName<< endl;
```

Lab Exercises

Lab Exercise 1 — Random Number Generator

The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
4. UML Diagram
3. Sample Output
4. Program Template
5. Problem-Solving Tips

The program template supplies 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

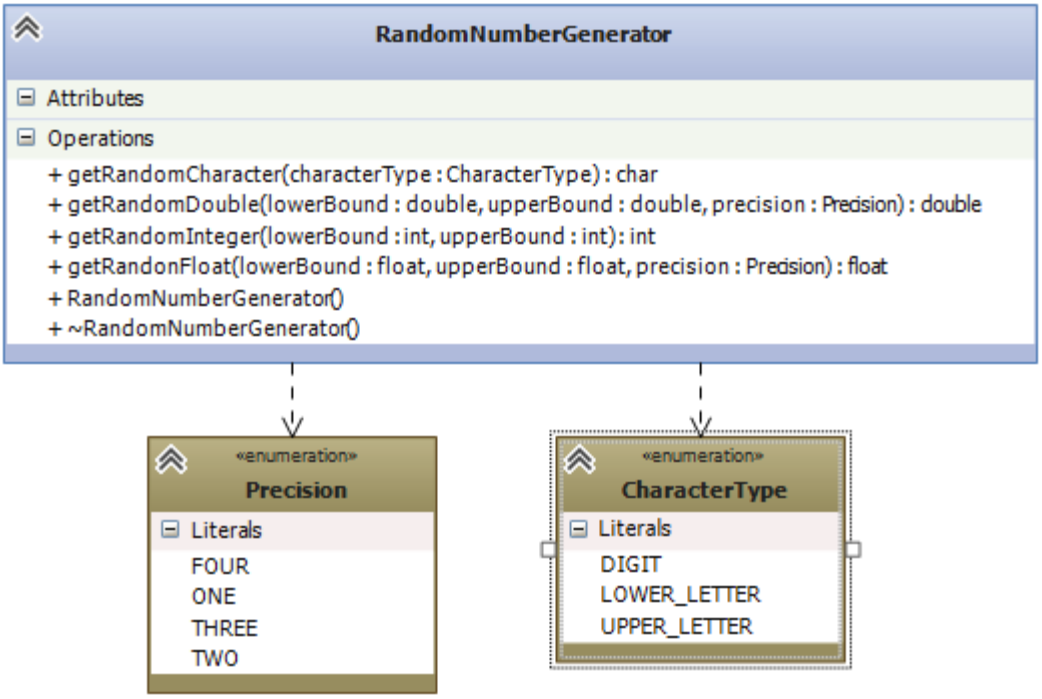
In this lab, you will practice:

- 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



Sample Output

```
+---------------------+
| Random Integer Test |
+---------------------+
SUCCESS : 13
SUCCESS : 19
+--------------------+
| Random Float Test |
+--------------------+
SUCCESS : 16.1
SUCCESS : 18.27
SUCCESS : 16.621
SUCCESS : 7.1123
+---------------------+
| Random Double Test |
+---------------------+
SUCCESS : 14.3
SUCCESS : 8.78
SUCCESS : 10.027
SUCCESS : 6.8886
+-----------------------+
| Random Character Test |
+-----------------------+
SUCCESS : p
SUCCESS : X
SUCCESS : 8
Press any key to continue . . .
```

Template

```
/// RandomNumberGenerator.h

#pragma once

class RandomNumberGenerator
{
public:

	enum Precision { ONE,TWO,THREE,FOUR	};
	enum CharacterType { UPPER_LETTER,LOWER_LETTER,DIGIT };

	RandomNumberGenerator();
	virtual ~RandomNumberGenerator();
	int getRandomInteger(int lowerBound, int upperBound);
	float getRandomFloat(float lowerBound, float upperBound, Precision precision);
	double getRandomDouble(double lowerBound, double upperBound, Precision
precision);
	char getRandomCharacter(CharacterType characterType);
};


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

RandomNumberGenerator::RandomNumberGenerator()
{
	srand(time(NULL));
}
RandomNumberGenerator::~RandomNumberGenerator()
{
	// Implement the function
}
int RandomNumberGenerator::getRandomInteger(int lowerBound, int upperBound)
{
	// Implement the function
}
float RandomNumberGenerator::getRandomFloat(float lowerBound, float upperBound,
Precision precision)
{
	// Implement the function
}
double RandomNumberGenerator::getRandomDouble(double lowerBound, double upperBound,
Precision precision)
{
	// Implement the function
}
char RandomNumberGenerator::getRandomCharacter(CharacterType characterType)
{
	// Implement the function
}
```

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

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;
	}
}
void TEST_RandomFloat(RandomNumberGenerator& generator, float lowerBound, float
	upperBound, RandomNumberGenerator::Precision precision)
{
	// Implement the function
}
void TEST_RandomDouble(RandomNumberGenerator& generator, double lowerBound, double
	upperBound, RandomNumberGenerator::Precision precision)
{
	// Implement the function
}
void TEST_RandomCharacter(RandomNumberGenerator& generator,
RandomNumberGenerator::CharacterType type)
{
	// Implement the function
}
int main()
{
	RandomNumberGenerator generator;

	cout << "+----------------------+" << endl
		<< "| Random Integer Test |" << endl
		<< "+----------------------+" << endl;
	TEST_RandomInteger(generator, 5, 20);
	TEST_RandomInteger(generator, 2, 60);
	cout << "+--------------------+" << endl
		<< "| Random 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
		<< "| Random 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);
	cout << "+-----------------------+" << endl
		<< "| Random Character Test |" << endl
		<< "+-----------------------+" << endl;
```

```
        TEST_RandomCharacter(generator,
        RandomNumberGenerator::CharacterType::LOWER_LETTER);
        TEST_RandomCharacter(generator,
        RandomNumberGenerator::CharacterType::UPPER_LETTER);
        TEST_RandomCharacter(generator, RandomNumberGenerator::CharacterType::DIGIT);
        return 0;
}
```

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 algorithm.

Assume: Precision value is equal 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 Algorithm can be applied to double number generation.

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

Lab Exercise 2 — Vehicle

The problem is divided into six parts:

1. Lab Objectives
2. Description of the Problem
3. UML Diagram
4. Sample Output
5. Program Template
6. Problem-Solving Tips

The program template gives 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

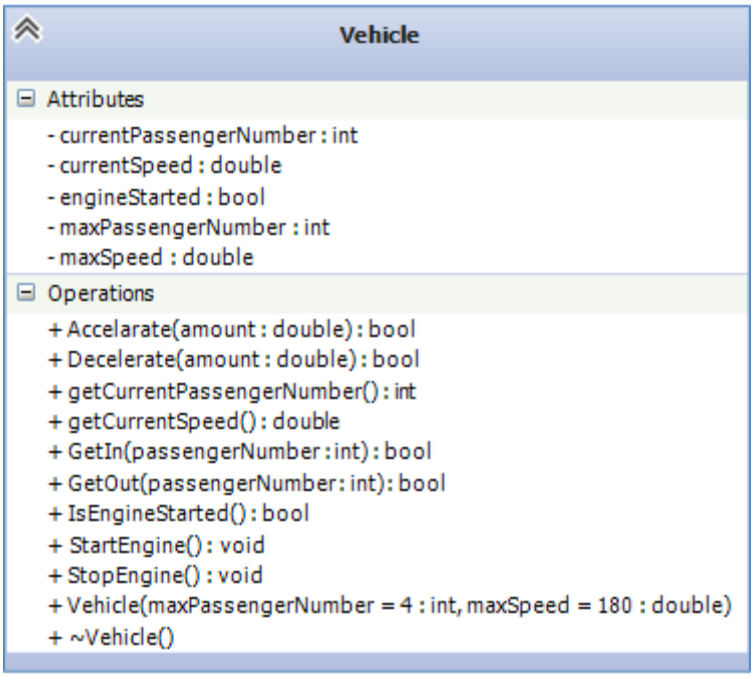
In this lab, you will practice:

- Declaring data members.
- Providing set and get functions to manipulate a data member’s value.
- Declaring member functions with parameters.

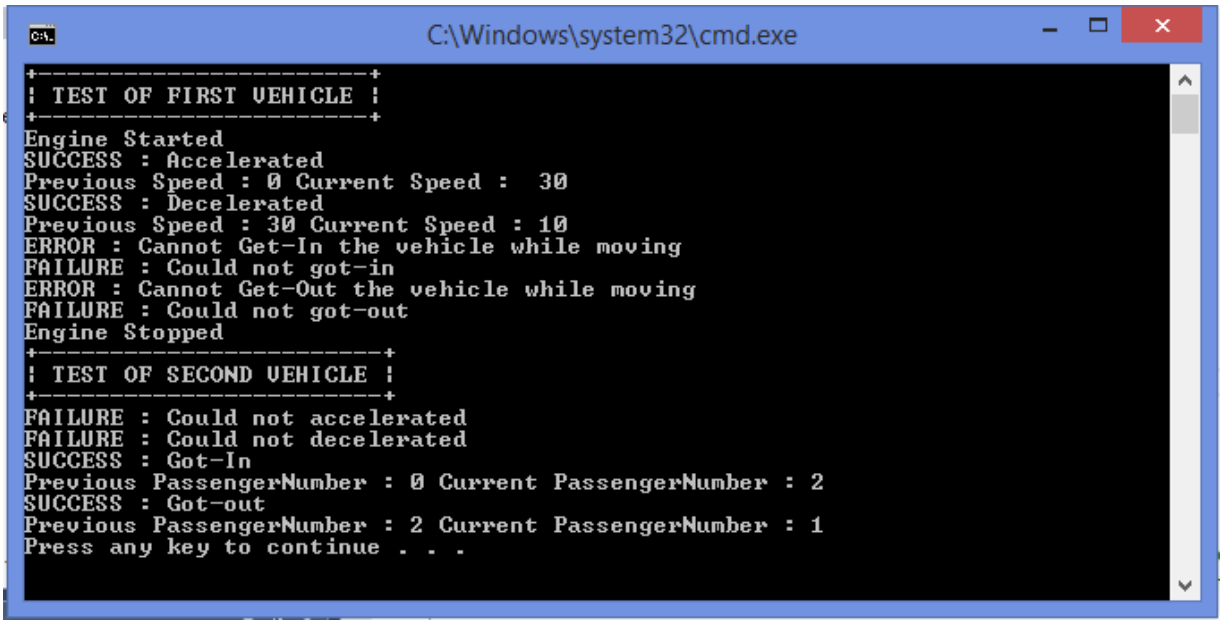
Description of the Problem

Write a Vehicle class that simulate a basic vehicle(automobile). 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



Sample Output



Template

```
/// Vehicle.h
#pragma once
#include <iostream>
using namespace std;
class Vehicle
{
public:

	Vehicle(int maxPassengerNumber = 4, double maxSpeed = 180);

	bool Accelarate(double amount);

	bool Decelerate(double amount);

	void StartEngine();

	void StopEngine();

	bool GetIn(int passengerNumber);

	bool GetOut(int passengerNumber);

	double getCurrentSpeed();

	double getCurrentPassengerNumber();

	bool IsEngineStarted();


private:
	bool engineStarted;
	double currentSpeed;
	int currentPassengerNumber;
	int maxPassengerNumber;
	double maxSpeed;

};
```

```
/// Vehicle.cpp
#include "Vehicle.h"


Vehicle::Vehicle(int maxPassengerNumber, double maxSpeed)
{
	this->maxSpeed = maxSpeed;
	this->maxPassengerNumber = maxPassengerNumber;

	this->engineStarted = false;
	this->currentSpeed = 0;
	this->currentPassengerNumber = 0;

	this->airConditionDegree = 22;
}


Vehicle::~Vehicle()
{
}


bool Vehicle::Accelarate(double amount)
{
// Implement the function
}


bool Vehicle::Decelerate(double amount)
{
// Implement the function
}


void Vehicle::StartEngine()
{
// Implement the function
}

void Vehicle::StopEngine()
{
// Implement the function
}


bool Vehicle::GetIn(int passengerNumber)
{
	// Implement the function
}




bool Vehicle::GetOut(int passengerNumber)
{
	// Implement the function

}
```

```
double Vehicle::getCurrentSpeed()
{
	return currentSpeed;
}


double Vehicle::getCurrentPassengerNumber()
{
	return currentPassengerNumber;
}


bool Vehicle::IsEngineStarted()
{
	return engineStarted;
}



/// VehicleTestMain.cpp
#include <iostream>
#include <string>
using namespace std;
#include "Vehicle.h"


void TEST_Acceleration(Vehicle& vehicle, double amount)
{
	double previousSpeed = vehicle.getCurrentSpeed();

	if (vehicle.Accelarate(amount))
	{
		cout << "SUCCESS : Accelerated " << endl;
		cout << "Previous Speed : "<< previousSpeed <<" Current Speed :  "<<
vehicle.getCurrentSpeed() << endl;
	}
	else
	{
		cout << "FAILURE : Could not accelerated" << endl;
	}
}


void TEST_Deceleration(Vehicle& vehicle, double amount)
{
// Implement the function
}

void TEST_GetIn(Vehicle& vehicle, int passengerNumber)
{
// Implement the function
}

void TEST_GetOut(Vehicle& vehicle, int passengerNumber)
{
// Implement the function
}
```

```
void TEST_StartEngine(Vehicle& vehicle)
{
// Implement the function
}

void TEST_StopEngine(Vehicle& vehicle)
{
// Implement the function
}

int main()
{
	cout << "+-----------------------+" << endl
		<< "| TEST OF FIRST VEHICLE |" << endl
		<< "+-----------------------+" << endl;

	Vehicle vehicle1(4, 220);
	vehicle1.StartEngine();
	TEST_Acceleration(vehicle1, 30);
	TEST_Deceleration(vehicle1, 20);
	TEST_GetIn(vehicle1, 2);
	TEST_GetOut(vehicle1, 1);
	vehicle1.StopEngine();


	cout << "+------------------------+" << endl
		<< "| TEST OF SECOND VEHICLE |" << endl
		<< "+------------------------+" << endl;
	Vehicle vehicle2(5, 180);
	TEST_Acceleration(vehicle2, 30);
	TEST_Deceleration(vehicle2, 20);
	TEST_GetIn(vehicle2, 2);
	TEST_GetOut(vehicle2, 1);


	return 0;
}
```

Problem-Solving Tips

1- Acceleration of the car depends on the engine status, if engine is not started, accelerate function returns false. Acceleration limit also depends on the maxSpeed of the vehicle. If any attempt to increase the speed of the vehicle to a higher value than maxSpeed, just set the currentSpeed to maxSpeed value.
2- GetIn and GetOut member functions depend on the current speed and passengerCount. If current speed is non-zero value. GetIn and GetOut functions returns false. If currentPassenger number is not enough to GetIn or GetOut, these functions print an error message and returns false. Otherwise, do the operation.
3- StartEngine and StopEngine functions print a message on the screen and set the bool member variable depending.
4- To implement the test functions, you can check the given sample test function.
5- Use ASCII table to find the corresponding random value ranges.