CSIT 840: Object-Oriented Programming in C++	Devvie Schneider Kent Assignment #2 (Due Date per Website)

Preface

This assignment is connected to ***Modules 5 and 6*** under the *Modules* link on the course ETUDES website. These modules also contain references to related textbook material.

Introduction

Many years ago but not too far away, at ancient Val E College, a malevolent instructor, oblivious to the fact that most students are strugging to make ends meet, decided to charge each of her students a fifty-cent toll to enter her classroom. There was a tollbooth just outside the door of each of her classes. Usually the students paid the toll, but sometimes a student successfully sneaked in without paying. (Those who were caught were punished with an "all you can eat" lunch buffet at the Val E College Cafeteria). The tollbooth kept track of the number of students that had gone by, and of the total amount of money collected.

With a number of courses per week and a number of students per course, this instructor needed a program to keep track of her ill-gotten gains. Therefore, she turned to her favorite source of free C++ programming labor, her CSIT 840 students.

As CSIT 840 students at "modern" Valley College, you will write the same program these ancient CSIT 840 students did.

What follows are the specifications and restrictions for this assignment. Also, be sure to carefully read the ***"IMPORTANT NOTES"*** given at the end of this document.

Class Members

Member Variables

Since we are all experts (or, at least becoming experts) on OOP, you will use a class, called ***TollBooth***, to represent the **tollbooth** (a ***single*** **course**). The following table identifies and describes the purpose of each *TollBooth* class member variable:

Member Variable	**Purpose**
m_nPayingStus	Number of students who paid toll for a course
m_nDeadbeatStus	Number of students who didn't pay toll for a course

m_nCourseCount	Number of TollBooth class instances ***initialized with input data***. Note that this is the only variable that applies to more than one course.

Restrictions*:** All of these class member variables must be **private**. You can **NOT** add to, or otherwise change, the declaration of these ***private class member variables. However, you do need to determine which are ***instance*** variables and which are ***static*** variables. (***Note*:** Some of these restrictions may appear to be somewhat artificial. Keep in mind, however, that I am also checking to see if you can implement certain concepts, such as the use of, and access to, a static variable. In other words, I am not *just* trying to torture you  , but there is method in my madness.)

Member Functions

You determine which class **member functions** you need, and whether or not they must be ***static,*** as well as whether or not they "must be" ***public*** or "could be" ***private***. **Note** that only a *static* member function will apply to more than one course; all other member functions are applicable to only a single course (and not an entire array of courses). ***In determining whether your class member functions are static or instance functions, and whether they could be private or must be public, you should take into consideration the code given to you below for the test drive***r. **Member functions are to be declared private, unless they must be declared public.**

Your project will have three files, ***toll.h*** (the declaration file for the *TollBooth* class), ***toll.cpp*** (the implementation file for the *TollBooth* class) and ***test.cpp*** (the "driver" file which tests the class).

Program Flow

1. Sample runs are reproduced below. As *Sample Run #1* below reflects, the program starts by asking the user to enter the maximum number of courses. In the same sample, the answer is 6 but only 4 courses are actually initialized with data. These numbers could be different. However, you may assume the answer is a positive integer.
2. The program then creates an *array* of *TollBooth* objects, each of which represents a **possible course**. Since the size of the array is determined by the user, you will have to create the array using *dynamic memory allocation*.
3. The program then asks two questions for each actual course to be initialized, how many students entered the room and how many students paid. You may assume that the user enters positive integers for both answers. The only input validation you need to be concerned with here would be if the user enters a higher number for paying students than for total students. In that case, you need to ask the user again to enter both the number of students who enter the room and the number of paying students. (See *Sample Run #2* below for examples of input validation.)
4. Once the two questions have been answered for each course, then the program outputs, ***for each course***, the following information, obtained the following way:

A. *The total number of students in the course* - Determined from accessing (through class member functions) the private class member variables that track the number of paying (*m_nPayingStus*) and deadbeat students (*m_nDeadbeatStus*) respectively. **Restriction:** ***Thus, the total number of students in each course cannot be derived from any variable declared in test.cpp; you are given no such variable in the test driver code.***

B. *The dollar amount of the toll collected* - Determined from accessing (through a class member function) the class member variable that tracks the number of paying students (*m_nPayingStus)* and the amount of the toll which is a constant fifty cents as stated above. (You may declare the toll amount as a symbolic or named constant in *toll.h*) **Restriction**: ***The number of paying students in the course cannot be derived from any variable declared in test.cpp; you are given no such variable in the test driver code.***

C. *The number of students who didn't pay* - Determined from accessing (through a class member function) the class member variable that tracks the number of non-paying students(*m_nDeadbeatStus*) . **Restriction**: ***The number of non-paying students in the course cannot be derived from any variable declared in test.cpp; you are given no such variable in the test driver code.***

5. Finally, the program outputs the average toll per course. The denominator in this calculation, the actual number of courses, is determined from accessing (through a class member function) the class member variable that tracks the actual number of courses initialized with data, *m_nCourseCount* . **Restriction:** T***he number of courses cannot be derived from any variable declared in test.cpp; you are given no such variable in the test driver code.***

A test driver file (test.cpp) is given next, with parts that you are to write indicated by comments. ***Restriction*****: Do NOT otherwise change the code that is given to you here**. You are given severl functions in addition to the main function that are defined in the test driver. ***Restriction:*** **You must include these test driver functions but you can NOT add additional functions to your test driver code.**

Below the driver code are some sample runs.

Driver File

The following code for *test.cpp* is given to you with comments that indicate where you must complete the code. You should **NOT** change the code for the ***main*** function and the other functions defined in the test driver that are given, except, of course, that you must write the statements that are indicated by comments. Also be sure that you do **NOT** violate the ***"restrictions"*** set forth in this assignment.

```
//test.cpp

/*

Insert all your include directives here
```

```
*/

void inputCourseData (TollBooth*, int);

void outputCourseInfo (TollBooth*, float&);

void displayAvgToll (float);


int main()

{

   int maxCourses; //input for dynamic memory allocation

   float totalTollAmt = 0; //calculated for all courses

   cout << "How many courses maximum? ";

   cin >> maxCourses;

   TollBooth* tboothPtr;

    //Here you must dynamically create an array of TollBooth objects, pointed to by tboothPtr


    inputCourseData (tboothPtr, maxCourses);

    outputCourseInfo (tboothPtr, totalTollAmt);

    displayAvgToll (totalTollAmt);

    //Here you should release your dynamically allocated memory

   return 0;

}


void inputCourseData (TollBooth* tbPtr, int max)
{
  char goOn = 'y';

  for (int i=0; i<max && goOn == 'y' ; i++)
   {
      cout << "Enter 'y' to initialize new course data; any other character to quit: ";
      cin >> goOn;
```

```
      goOn = tolower(goOn);
      if (goOn == 'y')
      {
         cout << "For course " << i + 1 << ":\n";
         tbPtr[i].setInputs();
        //(tbPtr + i)->setInputs();  //alternative
      }
   }
}

void outputCourseInfo (TollBooth* tbPtr, float& totalAmt)
{
   cout << fixed << showpoint << setprecision(2);
   for (int j=0; j<TollBooth::getCourseCount(); j++)
   {
      cout << "For course " << j+1 << ":\n";
      cout << tbPtr[j].getNumStudents() << " students are in the course\n";
      cout << "The toll collected is $" << tbPtr[j].calcTollCollected() << endl;
      totalAmt += tbPtr[j].calcTollCollected();
      cout << tbPtr[j].getNumDeadBeats() << " students didn't pay\n";
   }
}

void displayAvgToll (float totalAmount)
{
   if (TollBooth::getCourseCount() > 0)
      cout << "The average toll per course is $" << calcAvgToll (totalAmount) << endl;
   else
      cout << "No actual courses\n";
}

float calcAvgToll (float total)
{
   return total/TollBooth::getCourseCount();
}
```

Sample Run #1:

```
How many courses maximum? 6
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Enter 'y' to initialize new course data; any other character to quit: y
For course 1:
```

How many students entered the room? 5
How many students paid? 4
Enter 'y' to initialize new course data; any other character to quit: y
For course 2:
How many students entered the room? 6
How many students paid? 3
Enter 'y' to initialize new course data; any other character to quit: y
For course 3:
How many students entered the room? 7
How many students paid? 2
Enter 'y' to initialize new course data; any other character to quit: y
For course 4:
How many students entered the room? 8
How many students paid? 1
Enter 'y' to initialize new course data; any other character to quit: q
For course 1:
5 students are in the course
The toll collected is $2.00
1 students didn't pay
For course 2:
6 students are in the course
The toll collected is $1.50
3 students didn't pay
For course 3:
7 students are in the course
The toll collected is $1.00
5 students didn't pay
For course 4:
8 students are in the course
The toll collected is $0.50
7 students didn't pay
The average toll per course is $1.25

Sample Run #2:

How many courses maximum? 2
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Enter 'y' to initialize new course data; any other character to quit:
For course 1:
How many students entered the room? 5
How many students paid? 6
*** Invalid input: more paying than entered! ***
How many students entered the room? 5

How many students paid? 4
Enter 'y' to initialize new course data; any other character to quit:
For course 2:
How many students entered the room? 10
How many students paid? 12
*** Invalid input: more paying than entered! ***
How many students entered the room? 11
How many students paid? 12
*** Invalid input: more paying than entered! ***
How many students entered the room? 11
How many students paid? 11
For course 1:
5 students are in the course
The toll collected is $2.00
1 students didn't pay
For course 2:
11 students are in the course
The toll collected is $5.50
0 students didn't pay
The average toll per course is $3.75

Sample Run #3:

How many courses maximum? 4
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Now creating an uninitialized course element ....
Enter 'y' to initialize new course data; any other character to quit: x
No actual courses

***** IMPORTANT NOTES *****

- The variable *maxCourses* in *test.cpp* represents the number of elements in the dynamically created array of TollBooth (course) objects. However, this value may well be more than *m_nCourseCount*, the private member variable which keeps track of the actual number of courses initialized with valid user input data.
- The TollBooth class, and therefore the member functions of this class (with the exception of any static member function), deal with only one TollBooth (Course) object, not the entire array of TollBooth objects that the client application (test driver) declares.
- In the sample runs above, the message "Now creating an uninitialized course element ...." is displayed by the class when a TollBooth object (representing a course object) is initially ***created*** as an uninitialized element of the TollBooth array. Note that this message is not displayed by the test driver code.

- Questions to guide you in a proper implementation:
    - When is a constructor called for an array of potential class objects? Is it called automatically or can it be invoked at any time? How many times is the constructor called for the array?
    - What is the purpose of a static member variable? How many copies of the static members of a class are created? How is a static variable accessed? How is a static member function normally invoked? How is a static variable initialized?
- ***Remember***: As noted above, You should **NOT** change the code for the test driver that is given, except, of course, that you must write the code that is indicated by comments. Also be sure that you do **NOT** violate the ***"Restrictions"*** set forth in this assignment.

What You Turn In

Turn in to ETUDES (per the assignment submission instructions) a *zip* file of *toll.h*, *toll.cpp* and *test.cpp*. I don't need or want the other files that Visual C++ generates—just the code you wrote—as I can just include your files in a project I create. You also may include in your *zip* file a *pleaformercy.txt* file with any comments or message you want to give to me. Please sure to include your name in a **comment** at the top of each of your files (test.cpp, toll.h, and toll.cpp).

Need Help?

As stated in the Preface to this assignment, you'll find under *Modules* on the class' ETUDES website Modules 5-6. They should help you a lot. Please also review the sections of the textbook referenced in those modules.

You also can post in the *Assignment #2 Discussion Forum* on on the class' ETUDES website. Remember, you can post a few lines of code to provide context to your question, but please don't post all your code. As always, for reasons that should be obvious, please do not wait until the proverbial last minute to post.