3460:209 Assignment 4-A

Assignment 4-A: Who wants to be a millionaire

Overview

The purpose of this assignment is to make sure that you know how to write a program that contains functions that meet specific requirements. WARNING: Your work will receive no points if you solve the problem without implementing the three required C++ functions listed below (two versions of a C++ newBalance function and one version of a C++ yearsNeeded function). Moreover, your functions must work correctly for any valid arguments (not just for the particular numbers that your main program uses).

PROGRAM SPECIFICATION

For the assignment, write a program that an investor can use to compute future bank balances. The program asks the user for two pieces of information: The amount of an initial deposit (entered as a double number in dollars, such as 42.31 for 42 dollars and 31 cents), and an interest rate (entered as a double number, such at 0.05 for 5% interest).

The program prints the results of three computations, each of which assumes that interest is paid once per year on the amount in the account at the year's start:

(1) The balance in the account after one year of interest;
(2) The balance in the account after ten years of compounding interest;
(3) The number of years required for the balance to reach at least one million dollars.

The account balances should be printed with a dollar sign in front and rounded to two decimal places (such as $42.31).

In order to get numbers to print with two decimals, include both <iostream> and <iomanip> and place these lines in your main function (just after you declare any local variables that the main function uses):

```
cout.precision(DIGITS);
cout.setf(ios::fixed, ios::floatfield);
```

Your program must include functions with these exact prototypes:

```
double newBalance(double initialBalance, double interestRate);
 // This first version of the new balance function computes the
```

```
    // final account balance that will be reached by starting with
    // an initial balance and adding one year's interest at a given
    // interest rate (such as 0.05 for 5%). The function works
    // correctly for any non-negative arguments.

  double newBalance(double initialBalance, double interestRate, int n);
    // This second version of the new balance function computes the
    // final account balance that will be reached by starting with
    // an initial balance and adding some n number of years of interest
    // at a given interest rate (such as 0.05 for 5%). The interest is added
    // once per year and it is applied to the entire balance.
    // For example, new_balance(100.00, 0.10, 2) is 121.00 since
    // the first year received 10.00 interest (10% of 100) and the
    // second year received 11.00 interest (10% of the 110, which
    // was the second year's starting balance). The function works
    // correctly for any non-negative arguments.
    // Notice n is not an appropriate variable name, but will not be
    // deducted for this program.

  int yearsNeeded(double initialBalance, double interestRate, double goal);
    // The years needed function computes the number of years needed for
    // a given starting balance to reach a given goal at a certain rate
    // of compound interest. The function works correctly for any
    // positive arguments.
```

Important Note: Your program must include the three functions show above in this exact format. Validate all input. Do not use `using namespace std;`. We will actually test your program by throwing away your main program and using our own main program that calls the functions with lots of different numbers.

Make sure that your programs follow good documentation standards and have the same information required for each assignment. Reference the rubric standards on Brightspace.

Hints:

In the second version of newBalance, the balance after n years should be computed with the following formula:

```
initialBalance * ((1 + interestRate) raised to the n power)
```

Use the pow(x, b) function from <cmath> to compute the value of a number x raised to the b power.
The yearsNeeded function can be implemented by using logarithms. But if you are unfamiliar with logarithms, another approach is to repeatedly call newBalance with higher and higher years, until you get an answer that is at least equal to the goal. If you use this second approach, then start by calling newBalance with n=0, then n=1, then n=2, and so on. When you get a return value that is at least equal to the goal, then your answer is the current value of n.

Make sure that your programs follow good documentation standards and follow the requirements for assignments. Reference the rubric standards on Brightspace. Note functions and data validation are now required.

Submission Instructions – for programming solutions

On Brightspace, go to the matching Assignments for the **ASSGN@-#**, where @ is the chapter and # is the number or character of the problem assigned (eg., 5-11 for chapter 5, problem 11), and submit the program (cpp) and any (hpp) files.

Based upon material by Michael Main, University of Colorado Boulder.

Last updated 5.22.2016 by Will Crissey.

Be aware that programming falls under all of the rules of plagiarism. Be careful when using any coding found in the outside world that is not your own. Any evidence of plagiarism is subject to sanctions like forfeits, suspension, and even ejection, as determined by the Department of Student Conduct and Community Standards.