Interest Calculator
In this homework assignment, you will be writing a basic interest calculator to understand console input/output and basic arithmetic in C++.

Fixed Interest Calculator
You're going to write a small program that prompts the user to enter a principal amount (the starting amount), followed by the number of years, and then compute the total balance based on interest compounded once annually for that length of time. Don't worry I'll give you the interest formula. For now, we'll just use a **fixed rate of 5%**, but we'll change this later.

1. Create a new project, for example you could call it "homework02". **Do not use spaces in the name.** Follow the same process as you did in the first assignment; if you need a refresher, please refer back to Homework 1. Once you've created the project, open the "main.cpp" file in your project and delete everything in it.

2. We're going to need to include two header files for this assignment; one for text input/output and another for some more advanced math we'll be doing. So, put the following at the top of your file:

   ```
   #include <iostream>
   #include <cmath>
   ```

3. Write an empty `main()` function. Remember the `int` return type, and go ahead and have it `return 0;` at the end so you don't forget it later.

4. We're going to need a couple variables to store values that the user is going to enter. Inside `main()`, declare a variable of type `double` named `principal`, and declare another one of type `int` named `numYears`. You do not have to give them initial values.

   If you aren't 100% sure how to declare the variables please refer to the class notes, textbook, or in-class code examples.

5. Remember how we used `cout` to write text output to the screen? C++ also provides `cin`, which lets us read values that the user types at the keyboard. First, we want to ask the user to enter the principal amount, but before we use `cin`, we should output some useful text that describes what the user should enter. So, after the variable declarations above use `cout` to output a prompt like this:

   ```
   cout << "Please enter the principal: ";
   ```

   Notice that I included a space after the colon, and didn't put an `endl` at the end of the line. This means that when you run the program and are asked to enter a number, it will go on the same line as the prompt.

6. Next, you will use `cin` to store a number into the `principal` variable. We use `cin` a lot like we use `cout`, but we reverse the direction of the arrows to indicate which direction the data is traveling. After the line above, write:

   ```
   cin >> principal;
   ```

   In other words, what we are saying here is read a value from `cin` and store it in the variable `principal`. Since principal is a `double`, whatever the user types in will be treated as a decimal number (if possible).

7. Now repeat this process, asking the user for the number of years. Use `cout` to print an appropriate prompt then use `cin` to read a value into your `numYears` variable.

8. Now that you have your input values, declare a variable named `balance` of type `double` that will hold the final balance. Set `balance` to be equal to an expression that computes the balance based on the original principal and number of years, using 5% as the interest rate compounded annually.

   The mathematical expression for compounded interest is $B = P * (1 + r)^t$, where `B` is the final balance, `P` is the original principal, and `t` is the number of years, and `r` is the interest rate. **Note: The interest rate should be between 0 and 1; for example, 5% is 0.05.** Remember that the exponent should be computed with the `pow()` function.

9. Print the final balance using `cout`. Don't just print the number – print a meaningful message like "The final balance is " followed by the amount. Remember that you can chain strings and other values together by using << multiple times, so:

   ```
   cout << "The final balance is " << balance << endl;
   ```

   Would print the balance after that message.

10. Compile and run your program, and make sure it works as expected. You can test it with the following values: principal = 1000, number of years = 4. Remember: when you enter numbers larger than 1000, **do not type commas.** The final balance should be 1215.51, or close to it. Some computers might print the value out to a different number of decimal places, but that's ok.

Changing the interest rate

Now lets modify the program slightly before we finish up, so that it asks the user for the interest rate as well. This makes the program more general and flexible.

1. Declare a new variable called `rate` that will represent the interest rate as a `double`. It should be declared in the same place where you declared `principal` and `numYears`, (ideally at the top of the `main()` function).

2. **After the part of your program where you read the number of years** from the user, add another `cout` prompt and a `cin` line to ask the user for the interest rate. We want the user to enter the number as an integer percentage, without the percent sign and not as a decimal; in other words, for 5%, you would enter "5", **not** "0.05" or "5%".
3. Now, modify the formula that computes the balance to use the new `rate` variable instead of the fixed rate you used in Part I. In other words, change your `1 + 0.05` (or `1.05`, if you wrote it that way), to read `1 + rate / 100.`
4. Compile and re-run the program. Try it with the same values as before: principal = 1000, number of years = 4, rate = 5. Did you get the same result? If so, try it with another set of values just to be sure: principal = 7500, number of years = 10, rate = 9. The balance should roughly equal 17755.2.

What to Submit

For this assignment you should submit your `main.cpp` file from inside of your project directory.

This assignment will be graded automatically. Test your programs thoroughly before submitting them. Make sure that your programs produce correct results for every logically valid test case you can think of. Do not waste submissions on untested code, or on code that does not compile with the supplied code from the course website.

Web-CAT will assign a score based on runtime testing of your submission; your best score will be counted; the TAs will later verify that your best submission meets the stated restrictions, and assess penalties if not.

To submit this assignment:

1. Visit http://web-cat.cs.vt.edu in your web browser.
2. Enter your Virginia Tech PID and password in the appropriate fields on the log-in screen, and make sure that **Virginia Tech** is selected as the institution. Click **Login**.
3. The Web-CAT home screen will display useful announcements and assignments that are currently accepting submissions. Find the assignment that you want to submit in the table, and click the "Submit" button next to it.
4. Click the **Browse...** button and select the file you want to upload. The homework assignments and programming projects for this course should be self-contained in a single **main.cpp** file, so you can simply select that one file.
5. Click the **Upload Submission** button. The next page will ask you to review your selection to ensure that you have chosen the right file. If everything looks correct, click **Confirm**.

The next page will show that your assignment is currently queued for grading, with an estimated wait time. This page will refresh itself automatically, and when grading is complete you will be taken to a page with your results.

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the submitted file:

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants
//      assigned to this course.
//
//    - I have not used C++ language code obtained from another student,
//      or any other unauthorized source, either modified or unmodified.
//
//    - If any C++ language code or documentation used in my program
//      was obtained from an allowed source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
//    <Student Name>
```

Failure to include this pledge in a submission will result in the submission being disallowed during code review.