CIS 251 • C++ Programming
Laboratory Assignment 5: Functions

In this laboratory assignment you will use a function to calculate the discount rate to be applied to an order based on the quantity of items ordered. You will then use the value returned by the function to apply the discount to the order amount, which the program will then display with the discount to the user.

Background

The company requesting this program offers various discount rates based on the number of items a customer orders. The program the company wants is one that, given the number of items ordered and the cost per item, displays the order total with the appropriate discount rate applied.

Although the discount-determining logic could be written in the middle of the `main` function, these statements might fit better in their own function. Such a function accepts the number of items to be ordered as an argument to a pass-by-value parameter; it then uses a series of decision statements to determine the appropriate discount rate and returns this value to the `main` function. The `main` function then uses the returned value to calculate the order total with discount, which it displays to the user.

Step 1: Launch Visual Studio and Set Up an Empty C++ Project with Code

Open Visual Studio from the Start Menu. Start a new project of installed template / type **Visual C++**, subtype **Win32**, template **Win32 Console Application**, project name and location as desired (usually the Visual Studio **Projects** folder), and **Create directory for solution** box **unchecked**. In the **Win32 Application Wizard**, go to **Application Settings**, **check** the box for **Empty project**, and click **Finish**.

In the **Solution Explorer**, right-click **Source Files**, and select **Add > New Item…** from the shortcut menu. Select the category **Code** and the file type / template **C++ File (.cpp)**. Enter the file name **FLast_Lab05.cpp** (with your **first initial** and **last name** replacing **F** and **Last**, respectively). Click **Browse…** to select the location where you want to save your code (flash drive preferred).

Step 2: Begin Your Code

Add a single block comment or a set of line comments including your name, the course number, the laboratory assignment number, the date (optional), and a brief description of the program's contents. Place each of these items on a separate line. As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation, or you may wait until after you have written your entire program to add comments.

Next, enter the standard items that are part of your program: the `include` and `using` directives, the header for the `main` function, the opening curly bracket, a blank line after the bracket, the `return` statement, and the closing curly bracket.

Step 3: Write the Prototype for the Other Function

A function prototype (also called a function declaration) is a single statement that indicates to the compiler how a function is set up so it can verify the syntax of each call to the function. On a line between the `using` directive and the header for the `main` function, write the **prototype** for a function with the following attributes:

- Function Name: `getDiscountRate`
- Parameters: one parameter of type `int` to receive the quantity of the order
- Return Type: `double`

NOTE: A function prototype ends with a semicolon; the header of a function definition does not. See line 4 of Display 4.3 in your textbook, or your chapter 4 notes, for an example of a function prototype.

Step 4: Write the Definition for the Other Function

Although a programmer can write the code for the `main` function before writing those for other functions, it's usually best to know what the other function is doing before writing the statements in `main` that make use of it. On a new line below the closing bracket of the `main` function, begin by writing the **header** of the second function (almost identical to the function prototype).

WARNING: Do **not** put a semicolon at the end of a function header! The semicolon there would be an indication that the header is a prototype; the body would not be connected to this header.

WARNING: Although you have the option of omitting the parameter names in the function prototype, you **must** include names for each parameter in the header of the function definition.

After an opening curly bracket, **declare** a variable of type `double` to hold the value of the discount rate.

WARNING: You **cannot** give a variable the **same name** as a function.

Then, use a series of `if` statements to assign an appropriate value to this variable according to the following table:

Quantity	8 or less	9 to 12	13 to 25	26 or more
Discount Rate	0%	10%	15%	20%

HINT: The parameter variable receives the quantity when the function is called. You do not have to assign a value to the parameter in the function itself; statements that include the parameter use the value received from the function call.

WARNING: Percentages must be written in decimal form when used in calculations. For example, 20% would be written as 0.20 (or 0.2).

Finally, the function must include a `return` statement to send the value of the discount rate variable back to `main`. Write this statement, and then write a closing curly bracket for the function. See lines 31 through 38 of Display 4.3 in your textbook for an example of a complete function definition.

Step 5: Declare Your Variables in the `main` Function

The following table describes the variables needed in your `main` function (one variable per table row):

Description	Data Form	Initial Value
Quantity of items ordered	Whole number	From input
Price per item	Real number	From input
Discount rate	Real number	Returned from function
Total before discount	Real number	Calculated
Discount amount	Real number	Calculated
Total after discount	Real number	Calculated

Add a **declaration** for each variable after the opening curly bracket for `main`.

NOTE: It is possible to condense the calculations such that fewer variables are required; however, intermediate work variables help to clarify the arithmetic performed that contributes to the final result.

Step 6: Obtain Input from the User

At this point, the `main` function needs to prompt the user to enter the quantity ordered and the price per item:

```
Company Order Amount Calculator

How many items does the customer want?  (user enters value here)
How much does each item cost?  $(user enters value here)
```

Use a combination of **output** and **input** statements for this interaction.

Step 7: Call the Other Function

The discount rate will be determined by the second function. Write a statement that **calls** the second function, passing the variable storing the quantity ordered as the only argument, and assigning the returned value to the variable declared in `main` for the discount rate. Line 18 of Display 4.3 demonstrates a function call.

WARNING: A function call is a single statement. Don't write the entire function definition here!

WARNING: The name of the argument in a function call must be a variable declared in the same scope as the call. Do not use the parameter name as the argument name unless their names are identical.

Step 8: Perform the Calculations

Once the function returns the discount rate, the program can perform the following calculations:

total before discount = quantity of items ordered * price per item
discount amount = total before discount * discount rate
total after discount = total before discount – discount amount

Write **assignment statements** that use the variables you declared at the top of `main` to perform these calculations.

WARNING: If the spellings of the variables in the calculation statements do not match the spellings used earlier in your program, the code will not compile! Be consistent in your spelling!

Step 9: Display the Results

Display the three calculated values in a format similar to the following, remembering to include the numeric **formatting statements** before these **output statements**:

`Total Before Discount:` $*total before discount variable*
`Discount Amount:` $*discount amount variable*
`Total After Discount:` $*total after discount variable*

Step 10: Build Your Project and Run the Executable

Select the item **Build Solution** from the **Build** menu. If the **Output** area below your code indicates that there are syntax errors, double-click each line that describes a syntax error to see the statement that contains the error (or, in some cases, the statement after the error). Once you receive a message in the Output area that the build succeeded, select **Start Without Debugging** from the **Debug** menu, and enter some sample data for each choice in the console. You will need to select the Start Without Debugging command several times to test the program with different data sets.

Step 11: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description (especially the statements related to functions). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

In Blackboard, return to the Assignments page for the current assignment. Under the **Submission** section, click **Add Attachments**. Click **My Computer**; then, click the first **Browse…** button. Navigate to where you saved the source code file **FLast_Lab05.cpp** (with your name), select it, and then click **OK**. Verify that your file appears in the Submission section, and click the **Submit** button at the bottom of the page.

WARNING: If you do not click the Submit button, your attachment will not be saved!