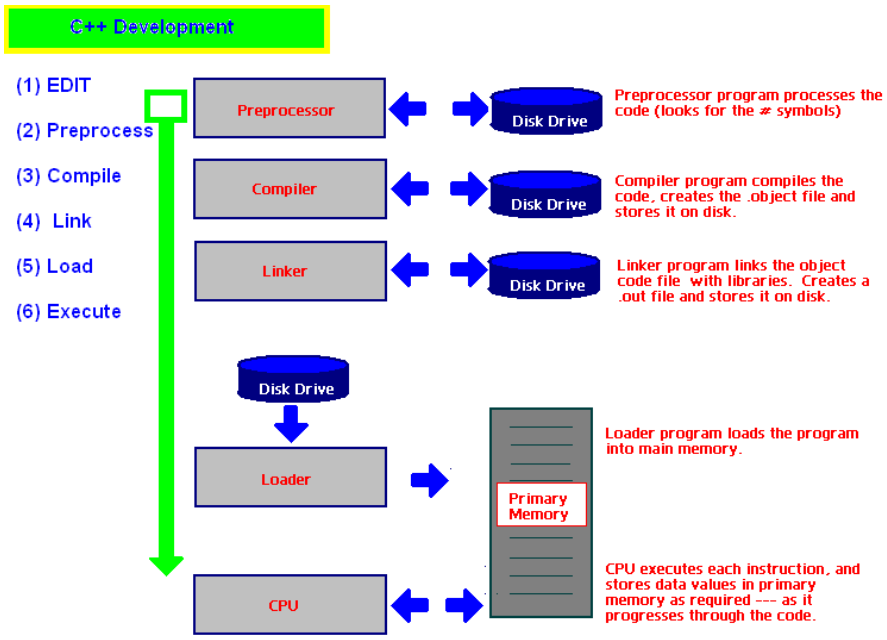
Lab 1 Due Date: Friday, September 4th, 2015 before 11:59 p.m

CS 150 **Lab Assignment 1** **Fall 2015**

Description: Implement problem solving skills through problem analysis, examination of source code examples, and problem solving. Learn about basic c++ concepts including tokens, key words, data types, statements, variable declarations, I/O operations, arithmetic expressions, and conditional blocks.

Before starting this lab: Read all of Chapter 1, and all of Chapter 2 in the Malik textbook. Once you have read (and re-read) the first two chapters, read through this entire assignment.

The graphic provides an abstract view of the "Processing a C++ Program" section of chapter 1. Study it, and be sure that you understand the steps involved, and what happens.



The following section in the text covers "Programming with the Problem Analysis-Coding-Execution Cycle". Throughout this course, you will learn problem-solving techniques, along with skills necessary to implement your problem solutions in C++. Most of our chapters contain problem analysis, algorithm design, and programming example source code in C++ - as a solution to the given problem. Note the errors shown in the graphic in this section of the textbook. How does your text describe syntax errors? What do you do to resolve syntax errors? The author uses an analogy to a recipe, in discussing algorithms. That is, a recipe is designed to complete a specific task in a defined number of steps (an algorithm!). Keep this in mind, as we examine a series of source code files in this lab – and note the steps involved in creating solutions (algorithms or 'recipes') to these simple programming example problems. As an extension to the analogy, consider that you cannot learn to be a good cook- without actually cooking! The same is true with programming. We will devote considerable time and effort toward examining & testing source code examples, and also toward creating our own source code solutions to problems. We will practice (cook). The examples for this lab, are taken from the textbook and should look familiar to you, as they follow the topics from start to finish in Chapter 2. Read and study the chapter repeatedly until these new concepts make sense to you.

Lab 1.1 Exploring the basic elements of a C++ program. statements, endl, cin, cout, escape sequences.

Lab 1.2 Identifying Data Types, and Reserved Words.

Lab 1.3 Arithmetic operations, operator precedence, type casting, and the string data type.

Lab 1.4 Running and examining C++ programs. Pre-processor directives. Good program style.

Lab 1.5 Writing simple programs, Solving simple problems, Simple Algorithm Design.

Lab 1.1 In this part of the lab, we will become familiar with the 'tokens' (reserved words, special symbols, and identifiers) used in a C++ program. Indicate whether the following representations is a reserved word (Appendix A), special symbol, or an identifier, or is invalid. Use chapter 2 as a reference, along with Appendix A.

TOKEN	Special Symbol	Reserved Word Symbol	Valid Identifier	Invalid
1. SALARY				
2. cat!				
3. float				
4. void				
5. r2d2				
6. 2hot4u				
7. %				
8. 7days				
9. @home				
10. s_u_m				
11. pattern				
12. +				
13. do				
14. <=				
15. false				

Lab 1.2 In this part of the lab, we review the simple data types found in a c++ program.

Value	Boolean	Integral	Floating Point	Character	Invalid
1. 3					
2. -6987					
3. 4.51					
4. false					
5. 'F'					
6. 9.4E4					
7. 3,345					
8. tralse					
9. 'u'					
10. 451					
11. 0.0					
12. true					
13. '+'					
14. 5.667					
15. 'off '					

16. The implicit conversion of one data type to another data value is called _____.
17. The explicit conversion of one data type to another data value is called _____.
18. The following is an example of the string data type: **"77"** True False
19. The following is an example of the string data type: **false** True False
20. The following is an example of the string data type: **"Hello ODU"** True False

Lab 1.3 In this part of the lab, we review the arithmetic operations, operator precedence, type casting, and the string data type concepts covered in chapter 2. Indicate the result of the following expressions.

	Expression	Result
1	20 % (4 - 2)	
2	20 / 4.0 * 6.4 /2	
3	4 % 5	
4	3.0 * (6 /24)	
5	3 - (3 + 3.0) * 10/3	
6	12 % 4	
7	7 * 6 / 21 /3.0	
8	11 % 4 * 3.0	
9	17 % 4 /3	
10	5 % 6 /4	
11	static_cast<int> (3.7)+5.3	
12	static_cast<int> (3.7 + 5.3)	
13	static_cast<double> (5/2)	
14	static_cast<char> (65)	
15	static_cast<float> (4)/3	

Lab 1.4 In this part of the lab, we study source code examples in detail, and compile and run them in our IDE.

1. Source code files for this lab are included in the zip file:
   a. **Create a CodeBlocks project** called Lab01 (on your Z drive on the CS network).
   b. Open each of the example source code files in CodeBlocks.
   c. Examine the source code of the programs line-by-line. Read all of the comments. Instructions for completing parts of the labs may be contained in the source code and comments of the examples.
2. Compile and run the programs. In the table below, provide a brief description of what each of the example programs is demonstrating. (use your textbook, and the source code comments to help).

• Example2_1	
• Example2_3	
• Example2_4	
• Example2_4_Mod	
• Example2_5	
• Example2_8	
• Example2_9	
• Example2_13	

• Example2_17	
• Example2_18	
• Example2_19	
• Example2_22	
• Example2_26	
• Example2_29	
• Example2_44	
• Example2_45	
• Example2_Ex5	
• Example2_MakeChange	
• Example2_Convert	

Lab 1.5 In this part of the lab, you will work with some simple programs and submit three source code files via a link on your lab blackboard site.

(A) Open the file called file: first_last_lab1A.cpp

Modify the source code program, so that it produces output ***similar*** to what is required in Programming Exercise 1, at the end of chapter 2 in our textbook (shown to the right).

PROGRAMMING EXERCISES

1. Write a program that produces the following output:

```
*********************************
*    Programming Assignment 1   *
*     Computer Programming I    *
*         Author: ???           *
*   Due Date: Thursday, Jan. 24 *
*********************************
```

In your program, substitute `???` with your own name. If necessary, adjust the positions and the number of the stars to produce a rectangle.

Modify the example so that it resembles the output shown to the right, putting your name instead of mine, the assignment due date, and CS ODU info as shown. You will add this **programmer info code** to the top of each source code file solution that you submit for all lab assignments.

```
**************************************
*        Lab Assignment 1            *
*  CS150 ODU Computer Science        *
*       Author: C. Boyle             *
*    Due Date: Friday, Sept 4        *
*      first_last_lab1A.cpp          *
**************************************
```

Read through all of the comments, and make the required changes to this program. You will make it interactive, by outputting a message to the user and then capturing the input and storing it in a variable of the correct data type. You will save this file with the required naming convention, and submit it via blackboard. Run the program and examine the output. Where do these values come from? Initialize the variables in step 2, and run it again.

(B) Open the file named: first_last_lab1B.cpp

- This is programming exercise 5, from chapter 2. Look at that problem in your textbook, and/or at the copy of it at the end of this lab. Follow those instructions, and rearrange the statements in the program correctly. Once you have the program running, add the **programmer info code**, to the top of the output of this program also. You will save this file with the required naming convention, and submit it via blackboard.
- Example 2-30 in our textbook illustrates proper program style and form. Reference that section again, and examine the four source code examples provided for this lab, called (formatting_1.cpp, formatting_2.cpp,…..). All four versions contain the exact same source code, but are formatted differently. Which are properly formatted? Which are improperly formatted? Do they all do the same thing? What do you see, from this, as the benefit of proper style and formatting?
- There is an example called Convert Length in chapter 2 of our textbook. Review it now. The problem, the algorithm steps, and the program solution are described, detailed, and provided. Open the source code file for that example (Example2_Convert.cpp), and examine it. Note that this is the proper style, formatting, and comments (documentation) that is required for the third source code submission of this lab assignment. (So indent it correctly, put comments in their to personalize your solution with documentation.) The description of this third file to submit follows.

(C) You are to study an algorithm, and create a program solution for the programming exercise on Newton's law, at the end of chapter two (also shown at the end of this document). The name of the provided source code file is first_last_lab1C.cpp. You will submit this file for credit via the lab blackboard submission link. ***(Substitute your first initial for first, and your last name for last.)*** Read through the list of numbered comments that reflect the steps of the solution algorithm. Complete those steps with the tips I provide below to create your solution to the problem. I have added the **programmer info code** to the source code program for you – make sure that you edit with the proper info, and that you change the comments at the top of the source code also.

My initial algorithm is given below as a simple five step process of declaring variables, input, calculation, output, and exit. These steps are expanded upon in the source code file.

Step 1: Declare and initialize variables identified in the problem domain. I see four that we need. Two variables to hold the values of mass1, and mass2. One variable to hold the distance value. And a fourth to hold the results of our calculation – the force. The graphic to the right is an example of how to declare and initialize the values under step one in the provided file. You will need to add an additional variable (for **force**) of the same type (double) to the following three declarations. Note that they are all initialized to a known value. Remember to properly initialize all of your variables.

```
double mass1 =0;
double mass2 =0;
double distance =0;
```

Step 2: Get the input values (Prompt the user for the values for both masses, and the distance value). I broke this down into multiple steps in the source file, so that we actually end up with ten steps to implement the entire algorithm. Examine that code, and note how this step of getting the input values is broken down into multiple steps. It's a simple series of statements that prompts the user for information, and stores the input into the appropriate variables. We can define multiple steps that achieve our goal of getting the input values, or we can do all this in one foul swoop. There's a flexibility here in defining the level at which we want to define our algorithm steps w.r.t. our level of understanding. The statements for steps 2 thru 7 are shown here. Copy them into the source code under the appropriately numbered comments.

```
cout << "\nEnter the mass of the first body: ";
cin >> mass1;
cout << "\nEnter the mass of the second body: ";
cin >> mass2;
cout << "\nEnter the distance between the 2 bodies: ";
cin >> distance;
```

Step 3: Make the required calculation. Use this statement.

```
force=(6.67/100000000)*(mass1*mass2/(distance*distance));
```

Step 4: Outputting the results, and Step 5- Exiting the program are complete.

Lab Grading Notes:

- All submitted program source code solutions – must run, without error to receive credit.
- All submitted program source code solutions – must contain the **programmer info code** to receive credit.
- Proper formatting, indentation, and documentation are required. Lose credit otherwise.
  NO late submissions for labs, under any circumstances. Upon the due date, the solution will be made available.

Newtons Law Problem from chapter 2 in our textbook.

Newton's law states that the force, F, between two bodies of masses M_1 and M_2 is given by:

$$F = k\left(\frac{M_1 M_2}{d^2}\right),$$

in which k is the gravitational constant and d is the distance between the bodies. The value of k is approximately 6.67×10^{-8} dyn. cm^2/g^2. Write a program that prompts the user to input the masses of the bodies and the distance between the bodies. The program then outputs the force between the bodies.

Programming exercise 5, from chapter 2

Consider the following C++ program in which the statements are in the incorrect order. Rearrange the statements so that it prompts the user to input the radius of a circle and outputs the area and circumference of the circle.

```
#include <iostream>
{
    int main()

    cout << "Enter the radius: ";
    cin >> radius;
    cout << endl;

    double radius;
    double area;

    using namespace std;

    return 0;

    cout << "Area = " << area << endl;

    area = PI * radius * radius;

    circumference = 2 * PI * radius;

    cout << "Circumference = " << circumference << endl;

    const double PI = 3.14;

    double circumference;
}
```