Honors Computer Science C++
Mr. Clausen
Program 6A, 6B, 6C, & 6G

Special Note: Every program from Chapter 4 to the end of the year needs to have functions!

Program 6A: Celsius To Fahrenheit Or Visa Versa Part 2 (10 additional points)

1. You are going to add some lines of code to your program to convert a temperature from Celsius to Fahrenheit or from Fahrenheit to Celsius. Save the program as LastNameFirstNameP6A.cpp in your "S:" directory.

2. Add a do…while loop to your **int** main function that repeats the entire program until the user enters a lowercase 'q' or uppercase 'Q'. The reserved word 'do' should appear after all of the variable declarations and initializations. The while statement will be near the end of the **int** main function and needs to have a compound Boolean expression to consider both cases of upper case Q, and lower case q. **Be sure to save your program before you run it, as you might end up with an infinite loop if your logic is not correct.**

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 6A folder.

PROBLEM 6B: Numbers, Their Squares, and Their Cubes 25 points

Write a program that uses a "for loop" that will ask the user for an integer between 1 and a user specified limit (I will test your program using the number 25) and print a list of these numbers, their perfect squares and their prefect cubes.
Save the program as LastNameFirstNameP6B.cpp in your "S:" directory. To see a model for this program look at the source code for the program "pizza.cpp" or "pizza2.cpp" in the network directory titled: HonorsCompSciCFiles. Look in the folder, Text Book Programs and Ch4.
As you type all your programs this year, be sure not to type past the 80-column line in Borland C++ 5.02 for Windows. If you have any statements longer than 80 columns, press the return key to "wrap" the statement around to the next line.

1. Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Make sure to change the program name and program description in these comments, so that the program number, name, and description say what is listed above.**

2. Include <iostream.h> so we can use the **cout** and **cin** commands. Also include <conio.h> so you can use **getch**() to leave your output displayed on the screen until the user presses any key to continue. Include <iomanip.h> so we can use the **setw** command. To raise one integer to another integer power you will need to add: #include <math.h>.

3. There are no constants to declare.

4. Don't forget to include comments for your functions, right before each of your function declarations.

5. Declare the function Display_My_Info ( ) in the function declaration section of the program. This should be a void function that doesn't take any parameters and doesn't return any values. In addition, declare functions for Get_Data(),and Display_Output(). The Get_Data() function should be a void function using a reference parameter of type **int** for the number that the user enters as the "upper limit" for your loop. Display_Output() should be a void function using a constant reference parameter for upper limit, and display the table of numbers, squares, and cubes.

6. Use a comment line with **equal signs** to separate all of the above from the int main function. For example:
   //==========

7. Inside the **int** main() function, declare all of your variables and then call the functions. **Don't forget that value-returning functions are "called" in assignment statements, while void functions can be called just by writing the function name including any of the "actual" parameters.** Then tell the user to press any key to continue, have your getch() command, and your return 0 command.

8. After the **int** main() function, have another comment line of **subtraction signs** to separate the above from your function implementation lines below. For example:
   //----------

9. Implement all of the functions, separating each one from the other ones using comment lines of **subtraction signs**.

10. When you implement the function: Display_Output(), create a "heading" for these numbers and have the numbers listed in three columns on the screen. To format the numbers nicely use the setw command to set a width for each of the columns. You also want to right justify these numbers. This will require that you add #include <iomanip.h> to your preprocessor directives and use the following commands: cout << setiosflags (ios::right) ; and cout<< setw(10); To raise one integer to another integer power you will need to add: #include

<math.h>; This can be used in an assignment statement or in a cout statement.

11. After the last function implementation of every program, end your program with two comment lines of equal signs. This signifies the end of your source code. For Example:
    //==========
    //==========

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 6B folder.

PROBLEM 6C: Arithmetic Series 25 points

Write a program to add all the numbers from 1 to a user specified limit (I will test your program using the number 100) and display the sum. The output of the program should be the numbers 1 through the upper limit in the first column, and the running total of the numbers in the second column. Although this could be written with a "for loop", **I am requiring that you to use a "count controlled while loop" instead**.
Save the program as LastNameFirstNameP6C.cpp in your "S:" directory. To see a model for this program look at the source code for the program "pizza.cpp" or "pizza2.cpp" in the network directory titled: HonorsCompSciCFiles. Look in the folder, Text Book Programs and Ch4.
As you type all your programs this year, be sure not to type past the 80-column line in Borland C++ 5.02 for Windows. If you have any statements longer than 80 columns, press the return key to "wrap" the statement around to the next line.

1. Type comments at the beginning of the program to display your name and other information just like those used for program 2A. **Make sure to change the program name and program description in these comments, so that the program number, name, and description say what is listed above.**

2. Include <iostream.h> so we can use the **cout** and **cin** commands. Also include <conio.h> so you can use **getch**() to leave your output displayed on the screen until the user presses any key to continue. Include <iomanip.h> so we can use the **setw** command.

3. There are no constants to declare.

4. Don't forget to include comments for your functions, right before each of your function declarations.

5. Declare the function Display_My_Info ( ) in the function declaration section of the program. This should be a void function that doesn't take any parameters and doesn't return any values. In addition, declare functions for

Get_Data(),and Display_Output(). The Get_Data() function should be a void function using a reference parameter of type **int** for the number that the user enters as the "upper limit" for your loop. Display_Output() should be a void function using a constant reference parameter for upper limit, and display the table of numbers and the running totals.

6. Use a comment line with **equal signs** to separate all of the above from the int main function. For example:
   //==

7. Inside the **int** main() function, declare all of your variables and then call the functions. **Don't forget that value-returning functions are "called" in assignment statements, while void functions can be called just by writing the function name including any of the "actual" parameters.** Then tell the user to press any key to continue, have your getch() command, and your return 0 command.

8. After the **int** main() function, have another comment line of **subtraction signs** to separate the above from your function implementation lines below. For example:
   //--

9. Implement all of the functions, separating each one from the other ones using comment lines of **subtraction signs**.

10. When you implement the function: Display_Output(), create a "heading" for these numbers and have the numbers listed in two columns on the screen. To format the numbers nicely use the setw command to set a width for each of the columns. You also want to right justify these numbers. This will require that you add #include <iomanip.h> to your preprocessor directives and use the following commands: cout << setiosflags (ios::right) ; and cout<< setw(10);

11. After the last function implementation of every program, end your program with two comment lines of equal signs. This signifies the end of your source code. For Example:
    //==
    //==

The output should look something like this:

Number	Sum
1	1
2	3
3	6

etc… until the user entered upper limit.

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 6C folder.

Program 6G program First Animation 25 points

For this program, don't draw any pictures in the background of the graphics screen. Keep the background a solid color (it can be black or any other solid background color). Don't forget that you must use functions for this program, and that you cannot use any global variables. **Remember to use comments for your function declarations and to have Pre and Post Conditions for each function also.**
Your job is to animate an object (more than just a circle or rectangle, etc.) that you have drawn across the graphics screen. Keep your "object" simple and small like a small boat, car, plane, etc. **For animation, draw the picture in a foreground color, then draw the same picture in the background color to erase it. DO NOT USE cleardevice( ) to erase your animated shape!**
Your animation cannot be strictly horizontal or vertical movement, it needs to be a combination of both, or be a linear animation from point (x_1, y_1) to point (x_2, y_2). You can also add parabolic motion or other mathematical motions from using other formulas that you may know.
Use any type of loop that is appropriate to your animation.

Remember the formula to calculate the equation of a line between 2 points:

$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

Note:
When you use initgraph and tell the compiler where Borland's bgi module is **use:** "c:\\bc5\\bgi" an absolute directory reference, instead of relative directory reference (what the book uses… "c:..\\bgi") I will deduct points off of anyone who does not make this change. This will work much better in our networked environment, because it is an absolute reference to the location of the bgi. The notation c: .. Simply moves up a level in your current directory. Our programs are on the network, and the absolute reference works best (especially for me when I need to grade your programs: get the hint?).

Please use this graphics mode:
int graphdriver = VGA, graphmode= VGAHI;

When you are finished with your program, have tested it thoroughly to make sure that your calculations are correct, and are sure that you don't need to make any changes, then save your program in the "W" network mapping, and the Program 6G folder.