Programming Assignment #1

Learning objective: Upon completing this assignment, you should be able to implement a simple class, as well as gain a better understanding of the building and use of classes and objects. Writing a simple C++ program

- Creating a class
- Creating private and public sections of a class
- Implementing the class
- Declaring objects
- Multiple file compiles
- Character array manipulation
- Good programming practices

Description:

An equilateral triangle is a triangle whose sides are equal. If two equilateral triangles are "glued" together along a common side, this will form a diamond. You are to write a class called Diamond, using filenames diamond.h and diamond.cpp, that will allow the creation and handling of diamonds based on the above description, whose sides are integers in the range 1-39.

What does a diamond look like:

The diamonds are shaped using ASCII characters. The ones drawn using the computer will have some unique characteristics.

- The size of the sides are diamond are calculated by counting the characters. You will notice that the characters at the very top and bottom as well as the left and right side are counted twice.
- You will notice that there is a space between all of the fill characters and a space after the first Border character before the fill characters are printed and stop one character before the last border character on the line.
- The Area of a diamond is the size*size.
- The Perimeter of the diamond is simply 4*size

Sample Diamonds:

Diamond of Size 3 with a '#' Border and '*' fill character

```
  #
  #  #
#  *  #
  #  #
    #
```

Diamond of Size 2 with a '$' Border, notice the fill does not show up.

```
    #
  #  #
    #
```

Diamond of Size 6 with Border of '#' and Fill of '*'.

```
          #
        #  #
      #  *  #
    #  *  *  #
  #  *  *  *  #
#  *  *  *  *  #
  #  *  *  *  #
    #  *  *  #
      #  *  #
        #  #
          #
```

Programming Specifications:

1. The constructor for the Diamond class should have 3 constant parameters: an integer size (required), which is the length of a side; a border character (optional, with a default of `'#'`); and a fill character (optional, with a default of `'*'`). If the size provided is less than 1, set the size to 1. If the size provided is greater than 39, set the size to 39. The class will need to provide internal storage for any member data that must be kept track of.

2. There should be member functions **`GetSize`**, **`Perimeter`**, and **`Area`**, which will return the size of a side, the perimeter of the diamond, and the area of the diamond, respectively. The first 2 should return integer results. The `Area` function should return its result as a double.

3. There should be member functions **`Grow`** and **`Shrink`**, which will increase or decrease (respectively) the size of the Diamond's sides by 1, unless this would cause the size to go out of bounds (out of the 1-39 range); in the latter case, `Grow` and `Shrink` should make no change to the size.

4. There should be member functions **`SetBorder`** and **`SetFill`**, which each allow a new border or fill character (respectively) to be passed in as a parameter. There is a chart of ASCII characters in an appendix of the textbook. The characters that should be allowed for the border or fill characters are any characters from the `'!'` (ascii 33) up through the `'~'` (ascii 126). If an attempt is made to set the border or fill characters to anything outisde the allowable range, the function should set the border or fill back to its original default (the ones listed for the

constructor -- the border default is `'#'` and the fill default is `'*'`).

5. There should be a member function called **`Draw`** that will display a picture of the Diamond on the screen. You may assume that the cursor is already at the beginning of a line when the function begins, and you should make sure that you leave the cursor on the line following the picture afterwards (i.e. print a newline after the last line of the diamond). Use the border character to draw the border of the diamond, and use the fill character to draw the internal characters. Separate the characters on a line in the picture by a single space to make the Diamond look more proportional (so that the halves look more like equilateral triangles**). You may not use formatting functions *<iomanip>* like *`setw()`* to draw the diamond.** This must be handled with loops. (You will only print out the newline, spaces, the border character, and maybe the fill character on any given line).

6. Write and provide a member function called **`Summary`** that displays all information about a diamond: its size, perimeter, area, and a picture of what it looks like. When displaying the area (decimal data), always show exactly 2 decimal places. Your output should be in the exact same format as mine (seen in the linked sample run below)

7. A sample driver program (called `driver.cpp`) will be provided in the Blackboard Assignment site for this assignment. I strongly suggest you modify and add more test cases to the driver to thoroughly test your program.

   The output from the sample output from the sample driver program in the Blackboard Assignment site for this assignment. Your class declaration and definition files must work with my main program, as-is (do not change my program to make your code work!). You are encouraged to write your own driver routines to further test the functionality of your class, as well. Most questions about the required behavior of the class can be determined by carefully examining my driver program and the sample execution. **Keep in mind**, this is just a **sample**. Your class must meet the specified requirements listed above in the specification -- not **just** satisfy **this** driver program. (For instance, I haven't tested **every** illegal fill character in this driver program -- I've just shown a sample). Your class will be tested with a larger set of calls than this driver program represents.

8. **General Requirements**
   - No global variables, other than constants!
   - All member data of your class must be private
   - **Use the `const` qualifier on member functions wherever it is appropriate.**
   - You will need to use the `<iostream>` library for output. You may use the `<iomanip>` library for formatting your decimal output to two places, if you wish to use the parameterized stream manipulators, but you may **not** use `setw()` or other output formatting functions for drawing the actual diamond. You may use the `<cmath>` library
   - When you write source code, it should be readable and well-documented.
   - Your diamond.h file should contain the class declaration only. The diamond.cpp file should contain the member function definitions and NOT have the main routine embedded in it. .

Grading Criteria:

- The program compiles. If the program does not compile no further grading can be accomplished. Programs that do not compile will receive a zero.
- (25 Points) The program executes without exception and produces output. The grading of the output cannot be accomplished unless the program executes.
- (25 Points) The program produces the correct output.
- (25 Points) The program specifications are followed.
- (10 Points)The program is documented (commented) properly.
- (5 Points)Use constants when values are not to be changed
- (5 Points)Use proper indentation
- (5 Points)Use good naming standards