**CSCE 121(505-508)** **Homework 3**
Due 11:59 p.m., 2/10/14 [125 points]

This Homework assignment is meant to give you more experience writing C++ programs that use some of the fundamental operations and catch errors. Also, it is meant to give you additional exposure to grammars.

All work should be turned into CSNet by the deadline. **In addition, please bring a hardcopy printout of your code to class the next day.** For programs, you need to turn in only the source code (not object or executable code). Your code will be tested using g++: you are welcome to develop in Visual Studio, but please make sure your code also runs in g++.

0) Create a text file, README, in which you:
   a. State the Aggie Honor statement, or else explain why you cannot do so.
   b. List any resources used, outside the textbook and discussions with the Instructor, TA, or Peer Teacher
   c. List any known problems with the assignments you are turning in. For example, if you know your code does not run correctly, state that. This does not need to be a long explanation.
   d. For places where you may have done some additional work (e.g. if you did the extra credit problem), put in a brief summary detailing what you did.

1) [50 points] These questions all have to do with grammars, and require only written answers (not a program). As a reminder, a grammar is a set of rules that determine what are or are not valid constructs in a language.
   a. [20 points] Here is a simple grammar that describes part of a fictional windowing system:

```
BaseCommands:
        "Create" Winname ";"
        "Delete" Winname ";"
        SetParam ";"
  Winname:
        String literal
  SetParam:

        WinName "set" Parameters
        "Set all" Parameters
  Parameters:
        SingleParameter
        SingleParameter "and" Parameters
  SingleParameter:
      ParamName "=" Value
  ParamName:
        "Border Color"
        "Width"
        "Height"
        "X Coordinate"
        "Y Coordinate"
  Value:
        Integer literal
```

   i. Identify which ones are and are not valid BaseCommands according to the rules above. For those that are, draw a parsing tree (similar to the examples in section 6.4) to show how the following statements would be parsed, and for those that aren't, draw a parsing tree to highlight where the parsing breaks down. [5 points each]
      1. Delete MyWindow
      2. ThatWindow set Width = 300 and Height =200;
      3. Set all X Coordinate = 100 and Border Color = red;
      4. Aorgihaionag set Y Coordinate = 0;

   b. [30 points] Write a grammar that can be used to describe written English statements of truth and conditions. Statements should be a statement that is true, either that something is true or in the form of an If…then… statement. All statements should be followed by a period.

i. All statements should end in a period.
ii. A valid if-then statement will always start with "If" and contain "then".
iii. The "and" and "or" conjunctions will group together smaller phrases.
iv. The words "If", "then", "and", "or", and "not" will not be part of any valid phrase, with the exception that a "not" might be included.
v. You do not need to allow for double/triple/etc. negatives. That is, if someone says "not", it will only be in that portion of the expression once.
vi. Note that there should be a precedence of options here. Basically, a "not" takes precedence over an "and", which takes precedence over an "or", which takes precedence over an "If…then…".
    1. So, you would interpret "It is dry or it is not raining and it is cloudy." as (It is raining) or ((it is (not) raining) and (it is cloudy))
vii. The following are examples of valid statements:
    1. If the sun is shining then it is not raining.
    2. The sun is shining.
    3. It is not cloudy.
    4. It is cloudy or the sun is shining.
    5. It is not raining and today is Sunday.
    6. If it is cloudy or it is raining then the sun is not shining.
    7. Today is Sunday or today is Monday or today is Tuesday.
viii. The following are not valid statements:
    1. The sun is shining (no period)
    2. If the sun is shining. (no "then")
    3. It is not not cloudy. (double negative)
    4. I am tired and or I am awake. ("and" followed by "or")
    5. I am awake if my eyes are open. (incorrect use of "if", no "then")

2) [25 points]. Imagine that you have, for some strange reason, the following function that you want to compute. Write a program that reads in an input from a user, and outputs the value of this function.
    a. The function F takes in a positive integer, n, and produces an integer result.
        i. F(1) is defined to be 1.
        ii. If n is divisible by 3, the value of the function is that integer (i.e. F(n)=n).
        iii. Otherwise, if the integer is odd, then F(n) = F((n+1)/2).
        iv. Otherwise, F(n) = F(n-1).
        v. For example:
            F(8) = F(8-1) = F(7) (since 8 is not 1, is not divisible by 3, and is not odd)
            F(7) = F((7+1)/2) = F(4) (since 7 is not 1, is not divisible by 3, and is odd)
            F(4) = F(4-1) = F(3) (since 4 is not 1, is not divisible by 3, and is not odd)
            F(3) = 3 (since 3 is divisible by 3)
        vi. Here are the first several values:
        F(1)=1, F(2)=1, F(3)=3, F(4)=3, F(5)=3, F(6)=6, F(7)=3, F(8)=3, F(9)=9, F(10)=9.
    b. Read input until the user enters 0.
    c. Note: you should write your code so that once a number is read, a function is called to compute the value.
3) [50 points] Write a program that allows people to compute the following arithmetic functions/geometric calculations. *Each of these should be implemented in your code as a separate function.* The input should read in the name of the function (a string, as given below), and then a list of parameters (as given below, in the order stated), and output the value. (5 points for reading/writing, 5 points per function, 5 points for error handling):
    a. Read input continuously until the user enters "End".
    b. Here are the functions to implement:
    CircleArea – radius (computes area of a circle with that radius)
    CirclePerimeter – radius (computes circumference of a circle with that radius)
    RectangleArea – length width (computes area of a rectangle with that width/height)
    RectanglePerimeter – length width (computes perimeter of rectangle)
    TriangleArea – side1 side2 side 3 (computes area of a triangle given the lengths of sides)
    TrianglePerimeter – side1 side2 side3 (computes perimeter of a triangle)

SquareRoot – value (computes the square root of the value)
Power – value exponent (computes value^exponent, for a floating point value and an integer exponent). You may use a floating point exponent if you prefer to do so.

c. As an example, a line of input might be: “RectanglePerimeter 3.0 5.0”, and the output would be 16.0, or the input might be “Power 3.0 2” and the output would be 9.0.
d. You should detect and smoothly handle errors in the input. Do not end the program if the input is invalid, just print a message that there was an error and continue on.

4) [60 points **extra credit**] Add the functionality you implemented in part 3 to the calculator program (you may start with the calculator program from the book, or the one on the course webpage). You should allow people to use these computations just like any other number.
   a. The commands given above should be used as the name, followed by parentheses containing the parameters, separated by commas.
   b. So, a command such as “CircleArea(3.0)+RectanglePerimeter(2.0, 3.0);” would be a valid statement in the calculator.