Assignment DS1

Due Date: September 13

Purpose
In this program, you will revisit and review some of the funtastic features of C++. These include basic classes, a constructor, methods, input and output, and operator overloading. You will also need to document your program as described in class.

Problem
Computers are quite stupid. They can only deal with the binary (base 2) number system. However, you need a large number of digits to represent a value in binary. Computers use bases other than 2 to represent numbers, the two most often used being hexadecimal (base 16) and octal (base 8). We humans, however, are not particularly good at arithmetic in anything other than the decimal (base 10) system. In this assignment, you will build a library so that a programmer can manipulate octal numbers as easily as one can manipulate decimals in the form of integers.

You may have learned about binary numbers. Binary means that only the digits 1 and 0 are used. For example, the decimal value 5 is equal to the binary value 101, also denoted as $101_2$. Although it seems odd at first, everything in binary works the same way as in decimal, only it’s in base 2 instead of 10. For example, in decimal, when you add numbers, you have to perform a carry when you reach 10. The same holds true in binary when you reach 2. So $23_{10} + 2_{10} = 25_{10}$ in binary would be $10111_2 + 10_2 = 11001_2$. The thinking in octal is much the same, except now the digits 0..7 are used. Thus, if we have $19_{10} + 6_{10} = 25_{10}$, the equivalent in base 8 would be $23_8 + 6_8 = 31_8$. This is because $23_8 = 19_{10}$, $6_8 = 6_{10}$, and $31_8 = 25_{10}$. It’s all equivalent, just with different bases.

Input
There is no input per se in this assignment; rather, you should write a small driver program (i.e., `main()`) that uses all of the operators and functions described below and/or that your program can correctly compute. Your driver program is entirely up to you. I will write my own `main()` to test your library. Because of this, it is essential that you name and implement functions and operators exactly as described below. An abbreviated sample program looks like the following:

```cpp
int main() {
    octal oOne, oTwo, oThree;
    int iNum;

    cout << "Type in two octal numbers to add: ";
    cin >> oOne;
    cin >> oTwo;
    oOne = oOne + oTwo;
    cout << "Sum is: " << oOne << endl;

    iNum = 19;
    oThree = iNum;
    cout << "Octal is: " << oThree << endl;
    cout << "Its decimal value is: " << oThree.toDecimal() << endl;

    return 0;
}
```
Output
The output depends on the test program and what the user types in. For example, if the above were run, and 14 and 25 were typed in for oOne and oTwo, the output screen would look similar to:

Type in two octal numbers to add: 14 25
Sum is: 41
Octal is: 23
Its decimal value is: 19

Specifics
You must create a class called octal. This class must contain data members so as to store a octal number, along with member functions (methods) for the various functions and operators. The class should be able to store one octal value up to 11 digits long. The value is assumed to be an integer (no decimal points).
Public features supported by the octal class include the following:

- a default constructor and a copy constructor. The default constructor should set the octal value to 0. The copy constructor should copy the argument as a class instance.
- an overloaded >> operator that allows the user to type in a positive (only!) octal number up to 11 characters long. You may assume the input will be correct; that is, a valid octal number will be typed in.
- an overloaded << operator that should display the octal value only (i.e., no other output, including newline).
- an overloaded + operator. This operator should add two octal values, both represented by instances of the octal class, or add an octal instance to an integer, or add an integer to an octal instance.
- an overloaded = operator where the left operand is a octal instance and the right operand is an integer or both operands are octals.
- an overloaded += operator where the the left and right operands are both instances of octal objects.
- a method called toDecimal that returns the integer decimal equivalent of a given octal.
- an overloaded post-increment operator ++. This operator should add 1 to a octal instance. Implementing this will give you 5 extra points.

Notes
Work on one function/operator at a time. Start with the input and output operators, and make sure they work. Next, complete the toDecimal() method. This will give you confidence that you now how to convert your values from octal to decimal. Then add operators, one at a time.

Be sure to name classes/methods as specified, because I will test your library with my own driver. We’ll both cry if things don’t work right because you did not named things as described above.
Be sure to adequately test your program! Do not assume that because one test works, your library will always work. Create some test cases on paper first, then check your program’s results. The sample program above is not a sufficient test.

Include an introductory comment at the top of your program as described in class. Each method should also have a comment, including what it returns and the types of parameters that are passed.

Submission: Name your source code file with your first initial and last name, followed by DS1, as in mgousieDS1.cpp. Submit this file via Canvas. Submit hard copy of your code at the beginning of lab on September 14th.

...operator functions are a purely syntactic convenience meant to clarify, not obscure, code.

– Dewhurst and Stark, in Programming in C++, page 76