Assignment DS1

Due Date: February 6

Purpose

In this program, you will revisit and review some of the *fun*tastic 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). It turns out that computer designer Mikel Goozly decided to use duodecimal (base 12) rather than octal or hexadecimal in his latest design. 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 duodecimal numbers as easily as one can manipulate decimals. All values will be in the form of integers; no floating point values need be considered.

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 duodecimal is much the same, except now the digits/symbols $0..9, A, B$ are used. Thus, if we have $19_{10} + 6_{10} = 25_{10}$, the equivalent in base 12 would be $17_{12} + 6_{12} = 21_{12}$. This is because $17_{12} = 19_{10}$, $6_{12} = 6_{10}$, and $21_{12} = 25_{10}$. It's all equivalent, just with different bases. Now with duodecimal, we have base 12, which means that we need the values 0, 1, 2, ... , 8, 9 but also 10 and 11. Since each value should be represented by one symbol, we will use A to represent 10 and B to represent 11. Thus, a duodecimal number such as $A3B_{12} = 1487_{10}$ $(10 \times 12^2 + 3 \times 12^1 + 11 \times 12^0)$.

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:

```
int main () {
   duodecimal dOne, dTwo, dThree;
   int iNum;

   cout << "Type in two duodecimal numbers to add: ";
   cin >> dOne;
   cin >> dTwo;

   cout << "Sum is: " << dOne + dTwo << endl;
   cout << "dOne is: " << dOne << endl;
```

```
   dOne = dOne + dTwo;
   cout << "dOne now is: " << dOne << endl;

   iNum = 22;
   dThree = iNum;
   cout << "Duodecimal is: " << dThree << endl;
   dThree = dThree + 4;
   cout << "Duodecimal is: " << dThree << endl;
   cout << "Its decimal value is: " << dThree.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 A3 and 9 were typed in for `dOne` and `dTwo`, the output screen would look similar to:

```
Type in two duodecimal numbers to add: A3 9
Sum is: B0
dOne is: A3
dOne now is: B0
Duodecimal is: 1A
Duodecimal is: 22
Its decimal value is: 26
```

Specifics

You must create a class called `duodecimal`. This class must contain data members so as to store a duodecimal number, along with member functions (methods) for the various functions and operators. The class should be able to store one duodecimal value up to 10 digits long. The value is assumed to be an integer (no decimal points).
Public features supported by the `duodecimal` class include the following:

- a default constructor and a copy constructor. The default constructor should set the duodecimal 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!) duodecimal number up to 10 characters long. You may assume the input will be correct; that is, a valid duodecimal number will be typed in.
- an overloaded << operator that should display the duodecimal value only (i.e., no other output, including newline).
- an overloaded + operator. This operator should add two duodecimal values, both represented by instances of the duodecimal class, **or** add a duodecimal instance to an integer, **or** add an integer to an duodecimal instance; that is, integers and duodecimals can be added in any order.
- an overloaded = operator where the left operand is a duodecimal instance and the right operand is an integer **or** both operands are duodecimal.

- a method called `toDecimal` that returns the integer decimal equivalent of a given duodecimal.
- an overloaded post-increment operator ++. This operator should add 1 to a duodecimal instance. This operator is optional. *Implementing this will give you 5 extra points.*

Notes

Work on one function/method 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 duodecimal to decimal. Then add operators, one at a time.

Be sure to name operators/classes/methods as specified, because I will test your library with my own driver. We'll be sad 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: All of the source code should be in one file. 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 February 7^{th}. Don't forget to include the **signed** Wheaton Honor Code statement (as described in the syllabus).

...operator functions are a purely syntactic convenience
meant to clarify, not obscure, code.
– Dewhurst and Stark, in *Programming in C++*, page 76