LAB ASSIGNMENT 21.1

Numbers

**Background:**

Numbers with different base values occur quite often. Clocks have numbers with bases of 60 for minutes and seconds and 24 for hours. Angle measurement has a base of 360 for degrees and a base 60 for minutes and seconds. The number systems we have looked at in this lesson have bases of 10, 2, 8 and 16.

**Assignment:**

For this assignment, we will create a number class that can help us create classes for objects like clocks or angles. The number class will have to be general enough so that we can define different bases for it. Clocks need numbers of base 24 (or 12) and 60. Angles need numbers of base 360 and 60. For now, we are going to simplify things and only define constructors, a toString method, and an increment method. Of course, a complete number class would have many other methods, such as adding, subtracting, multiplying, comparing, and negating the numbers. Since a number is made up of digits, we will also define a digit class. The number and digit classes appear very similar because they have nearly the same methods. The differences are in the implementations of the methods.

Write a number class that represents a number with a given base. The number class has two constructors. The default constructor sets the base at 10 and the value to 0. The other constructor accepts a decimal value and a base. The constructors will have to create all of the digits (as objects of the digit class) and store them in some kind of list. Create a toString method and an increment method. These two methods will use the corresponding methods of the digit class to do most of the work for them.

Write a digit class that represents a single digit with a given base. The digit class has two constructors. The default constructor sets the base at 10 and the value to 0. The other constructor accepts a decimal value and a base. Create a toString method and an increment method. The increment method should return a boolean: true if there is a carry and false if not.

For both classes, you can assume that the bases are restricted to binary, octal, decimal or hexadecimal and the values are non-negative.

**Instructions:**

Write a test driver that proves that your classes work.