# EEN 218
# Lab #3
# Dynamic Allocation

In this assignment, you are to implement a calculator for use with very large numbers (integers in this case). The normal int used in C++ is a 32-bit number that has a limited size after which it overflows. In order to do some calculations, it is necessary to allow calculations on integers of arbitrary size.

For this purpose, you will build a class to support large unsigned integers.

**You need to provide a report that shows: (5)**

1. Your class designs
2. The algorithms used for increment, add, subtract, multiply, divide
    a. This should be in pseudo-code or flowcharts, not just the code you ended up writing
3. Compiling instructions
4. Sample runs (screenshots are fine)

**Part1: The basic object (30)**

1. Design a class BigInteger that allows the user to store integer values of any size (practically it will still be limited to the memory available and about 4-billion digits)
2. The class should store the digits for the integer as a dynamically allocated array of digits using pointers. This should be able to grow at any time.
    a. This is basically an implementation of a vector of single digits (char or int type, it's up to you)
    b. You must write the vector implementation yourself, you are not allowed to use the vector from the STL.
3. Make sure all the internal properties of the class are private
4. Have a constructor that sets the value of the internal integer representation from an input string so that we can have initializations similar to:

```
BigInteger b1("12345678901234567890");
-or-
BigInteger *b2;
b2 = new BigInteger("9876543210987654321");
```

5. Remember to include the no-parameter constructor
6. Remember to include a destructor
    a. Since you are using dynamic memory, you need to properly clean up any allocated memory.
7. Have a function to print out the BigInteger

**Part 2: The easy ones, increment, add, multiply (45)**

1. Write methods for the BigInteger
    a. At least, increment, add, multiply
        i. All functions are with positive integers only
        ii. Increment just increments the calling object
        iii. Add/Multiply passes one BigInteger as a reference parameter and adds/multiplies it to/with the calling object
    b. The operations may be as the following examples:

```
BigInteger b1("123");
BigInteger b2("111");

b1.add(b2);
b1.print(); // should result in 233 being printed
```

2. Test your code with some small examples.

**Part 3: Somewhat more difficult: subtract (positive results only) (20)**

1. Write a method for subtracting two BigIntegers
2. The method should pass one BigInteger as a reference parameter and subtracts that from the calling object
    a. The method should determine if a negative result will occur and return an error
    b. A Compare method is very useful in this case

**Extra Credit: The hard one: divide (integer divide) (15)**

1. Write a method for divide (this is the hard one)
2. More points will be awarded for an elegant and fast divide algorithm
3. Test your code with some examples