CSCE 121(505-508) **Homework 9**
Due 11:59 p.m., 4/15/14 [170 points]

This Homework assignment is meant to help you understand more about arrays and templates.

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, put in a brief summary detailing what you did.
   e. Include in your README file the short summary to the question in part 1.

1) [170 points] You are to write a template class for handling (univariate) polynomials. Polynomials are equations made up of several terms, where each term has a coefficient. For example, one polynomial would be $3x^3 + 5x^2 -2x -1$. The degree of the polynomial is the highest power term that has a nonzero coefficient. (note that many terms can be zero).
   - [25 points] Polynomials are often characterized by the type of their coefficients (int, float, double, for instance). Your polynomial class should use templates and allow someone to define a polynomial of any type.
   - [25 points] You will want to store the coefficients. **For this assignment, do not use a vector to store the coefficients!** I expect you to use arrays as we talked about when discussing the implementation of vector (i.e. there might be similarities between your coefficients and the vector implementation we looked at), though there are some other ways to do this, also. I expect you to allocate space for the coefficients using "new".
   - [15 points] You should let someone add a term to an existing polynomial, and adjust the space used for that polynomial as a result.
     - For example, one option to create the polynomial $3x^2 + 500$ might be:
       ```
       poly<int> p1;
       p1.addterm(500,0);
       p1.addterm(3,2);
       ```
   - [20 points] You should provide the ability to copy polynomials via the = command, and in a constructor.
     - For example, you should be able to have:
       ```
       poly<int> p1;
       // Do something here to set p1
       poly<int> p2(p1);
       poly<int> p3;
       p3 = p2;
       ```
   - [20 points] You should provide an output mechanism for polynomials using the >> operator. Use a reasonable text output, where the polynomial can be read/understood.
   - [15 points] You should allow polynomials to be added together
   - [10 points] You should allow polynomials to be multiplied by a constant integer
   - [20 points] You should allow polynomials to be multiplied together
   - Note that you might want to include additional routines to assist you in the above, or to make the class more complete (e.g. add subtraction).
   - [15 points] You need to include a main file (or other test) that creates polynomials and demonstrates the various operations clearly. Be sure to clearly demonstrate all of the required parts of the program.
   - **[This question answer should go into the README file!]** [5 points] Try creating a polynomial where the coefficients are not numbers – e.g. a character. Describe what happens, when you try to do this and why

that is occurring.

- [10 points extra credit]. Provide a routine to compute and return the derivative of a polynomial.
- [10 points extra credit]. Provide a routine to evaluate the polynomial at a particular value. In other words, if the user specifies the value for x, the function returns the value of the polynomial.

2) [100 points Extra Credit] Write a class to support rational numbers. Rational numbers are fractions (numerator and denominator).
   - [15 points] There should be some basic representation and assignment operations.
   - [25 points] You should define the +, -, *, and / operators for rational numbers.
   - [10 points] You should support copying via constructors or the = operator
   - [10 points] You should support output streams (i.e. <<)
   - [15 points] When representing numbers, use the simplest possible representation – i.e. remove common factors from the numerator and denominator. So, 15/12 should become 5/4.
   - [15 points] Add support for ==, !=, >, >=, < <=
   - [10 points] Use your rational number type in the polynomial class, above, to be able to have polynomials with rational coefficients.
3) [60 points Extra Credit] Create a program that can print out a Hadamard matrix of a given size.
   - A Hadamard matrix of size nxn is a matrix with entries that are either 1 or 0, and in which any two rows or columns differ by exactly n/2 elements.
   - You can form a Hadamard matrix as follows: A 1x1 matrix is a 1 (or choose 0 – it doesn't matter). Then, you can form a 2x2 matrix by making 3 copies of that first matrix, and putting them in the upper left, upper right, and lower left corners of the matrix. Take the exact opposite of that matrix, and put it in the lower right.
   - Your program should ask the user for a size, n for n>=0. The matrix should be of size 2^n. So, if the user enters 3, the matrix would be 8x8.
   - The program should then output a Hadamard matrix of the appropriate size.
   - For reference, here are examples of the first few Hadamard matrices:

```
1: 1
2: 1 1
   1 0
3: 1 1 1 1
   1 0 1 0
   1 1 0 0
   1 0 0 1
4: 1 1 1 1 1 1 1 1
   1 0 1 0 1 0 1 0
   1 1 0 0 1 1 0 0
   1 0 0 1 1 0 0 1
   1 1 1 1 0 0 0 0
   1 0 1 0 0 1 0 1
   1 1 0 0 0 0 1 1
   1 0 0 1 0 1 1 0
```