EECS 211 – Spring Quarter, 2010
Program 2

Due Wednesday, April 21st, 2010 at 11:59PM

You are to write a program which reads polynomial data (power and coefficients) for polynomials up to order 5 (i.e., highest power of x is 5, therefore 6 coefficients total counting the constant term) from a file, forms a second polynomial which is the derivative of the one read from the file, displays both the polynomial and derivative, and then displays a table of values of x, polynomial(x) and derivative(x) for the six values 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0. Be sure the output on the screen is neat and informative. The file contains data for several polynomials; each set starts with an integer n telling the highest power of x in that polynomial and continues with n+1 real numbers representing in order the constant term, the coefficient of x, the coefficient of x^2, and so on. Your program should prompt the user to enter 1 before going on to the next set so that the user can examine the output for each polynomial.

In this program you will write four functions.

- The first function should read the highest power of x and then read the coefficients. Remember, there will be one more real number than the highest power of x.) This function has one parameter, an array of float numbers. It has a return type of int and returns the value read for the highest power of x in this polynomial.
- The second function should display the algebraic form of a polynomial on the screen. It has two parameters - the highest power of x and an array of coefficients. The output should have the terms separated by the symbol "+" and each term appear with the coefficients followed by the correct number of "*x" characters. For example, if the polynomial is of order 3 and the coefficient array contains 1.5, 2.7, 3.1, 1.6, …, then the output should look like

  1.5+2.7*x+3.1*x*x+1.6*x*x*x

- The third function should compute the coefficients of the derivative polynomial. It should have three parameters - the highest power of x in the polynomial, the array of coefficients of the polynomial, and an array that will receive the coefficients of the derivative.
- The fourth function should have a return type float and should compute the value of a polynomial for a given value of x. The parameters are the highest power of x in the polynomial, the array of coefficients of the polynomial, and the value of x.

Hints and requirements

1. Include the fstream.h header file from C++ and the following line at the file level (i.e., NOT inside any function):

   ```
   ifstream inp("polydata.txt", ios::in);
   ```

   You may then use the variable inp to read from the file polydata.txt just like you would use cin to read from the keyboard.

2. Develop your program in pieces. This would be a good time to try top-down design and development. Here is an appropriate sequence of steps:
   a. Write just the main function. It does nothing more than loop until the user says to stop.
   b. Write the first function (read one polynomial). Add a call to this function in main. Have main print out the degree and then print the values in the array – just the values, no algebraic format yet. Test this much to see that you are reading the data properly.
   c. Then write and test the second function.
   d. Then perhaps add the derivative function and augment the basic loop in main() to print the derivative in addition to the original polynomial, and finally add the evaluation function and finish your main() function.

3. You may find it more convenient to develop you program by entering data from the keyboard. That allows you to try more variations of data and test cases without having to modify the txt file. Of course, you would then need to change so that the program you turn in takes its input from a file by the name polydata.txt

4. You should begin to use defined constants in your programs. I used three in my solution to this program - MAX_POWER, NUMBER_OF_COEFFICIENTS and DEBUG. I defined DEBUG to be 1 when I wanted to run the program and have the input come from the keyboard and 0 when I wanted to run it with the input coming from the file. I then used statements of the form

   ```
   if(DEBUG)
     {  …
     }
   else
     { …
     }
   ```
   where I wanted to switch back and forth between running the program with input from the keyboard vs from the file.