COP 3502 – Spring 2004 – Programming Assignment #4

Due Date: Tuesday March 30, 2004 by 11:55pm

Points: 45 points

Objective: The primary focus of this lab is to give you practice implementing and maintaining linked lists. All linked list implementations are to be dynamic.

Application: A list can be used to represent a polynomial. In general, a *n*th order polynomial in *x*, for non-negative integers *n*, has the form:

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \cdots + a_n x^n$$

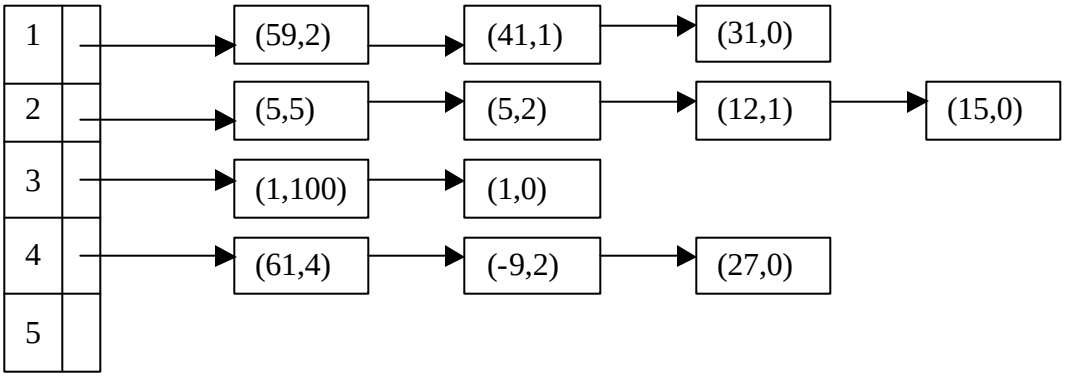
where: $a_n \neq 0$ and a_i is the coefficient of the *i*th power of *x*.

For simplicity, we will assume that the coefficients are integer values. An alternative representation of such polynomials consists of a sequence of ordered pairs:

$\{(a_0, 0), (a_1,1), (a_2, 2), \ldots, (a_n, n)\}$

Each ordered pair (a_i, i) corresponds to the term $a_i x^i$ of the polynomial. For example, the polynomial $31 + 41x + 59x^2$ would be represented by the sequence {(31, 0), (41, 1), (59, 2)}. The advantage of such a representation is that zero coefficient terms are not included in the sequence. This representation can result in major space savings when representing a 100^{th}-order polynomial such as $x^{100} + 1$ which includes only two terms in this representation: {(1, 100), (1, 0)}.

You will build singly-linked list structures that will represent polynomials using this representation technique. The input for each polynomial will consist of a sequence of ordered pairs (coefficient, exponent). You will have no more than 15 polynomials in total. One way of representing the polynomials is shown below. Each polynomial is uniquely identified by its position within an indexing structure (a simple array will work fine for this). This storage scenario is illustrated below:



When constructing a polynomial, the user may input the coefficient,exponent pairs in random order, however, the polynomial should be stored in decreasing order of exponent. The indexing structure is filled sequentially.

For the purposes of this application, we will assume that the "data" portion of the list node contains two integer values, the coefficient and the exponent. You can augment the Node class that we developed in lecture.

Operations: You will develop several different functions that operate on the polynomials. Each of these is explained below:

build: This function will construct a new polynomial. It will require the user to input the pairs which correspond to the coefficient and exponent pairs that comprise the polynomial. This new polynomial is simply appended to your list of polynomials. (Hint: This function should most likely return a pointer to the first node in the list storing the newly created polynomial.)

print (p): This function will produce a nice looking mathematical format for the polynomial *p* printed with the largest exponent term first (leftmost printed term) down through the constant term. (Don't worry about printing exponents in smaller print and proper position – use form shown below.) NOTE: this function does not alter the list representing the polynomial it simply prints it out in the proper order (as the polynomial is already stored).

example: print called on the fourth polynomial listed on the previous page will print: 61x^4 – 9x^2 + 27

differentiate (p): This function will produce the derivative of the polynomial with respect to *x*. The derivative is obtained by differentiating each term in the polynomial with respect to *x.* The derivative that you produce will be stored in a new polynomial (one of the 15 you have available).

Given the polynomial: $\{(a_n, n), ..., (a_2, 2), (a_1,1), (a_0, 0)\}$
The derivative with respect to *x* is: $\{(na_n, n-1), ..., (2a_2, 1), (a_1,0)\}$
(Note: If you need further clarification on the process of taking a derivative of a polynomial, please see your TA.)

example: differentiate called on the first polynomial listed on the previous page will produce the list {(118, 1), (41, 0)}.

(Hint: Your function should taken in a pointer to the first node in a polynomial, create a new linked list structure that stores the corresponding derivative, and then returns a pointer to the first node in this newly created derivative.)

end: When the user has finished they will indicate this (see Input below) and the system should print each polynomial currently in the system.

Input: The input to your program will be menu driven, based on the functions shown above. Your menu screen could look like the one shown below or something along these lines.

```
Welcome to the CS1 Polynomial Representation
                    System
             -Select your option-

   1.   Create a new polynomial.
   2.   Print an existing polynomial.
   3.   Differentiate an existing polynomial.
   4.   End session.

ENTER YOUR SELECTION:
```

The build function will request the user enter a set of ordered pairs representing the polynomial. The print and differentiate functions will request the user enter an integer number (between 1 and 15) indicating which polynomial is to be printed or differentiated. Notice that differentiate creates a new polynomial and will need to know the proper location in the list to insert the new polynomial.

Input Specifications and Error Checking: Assume that the user will choose proper menu choices and will enter all polynomials perfectly. Thus assume that the user properly enters the number of terms they are entering(a positive integer), and does NOT enter two terms with the same exponent. Furthermore, each exponent entered will be a non-negative integer and each coefficient entered will be a non-zero integer. However, a user may attempt to print or differentiate a polynomial that does not exist. In this case, simply print out an error message that states the polynomial doesn't exist and print out the main menu.

Output: Only the print and end functions actually produce output. You may, if you wish, have the build and differentiate functions print a message that they have completed their task.

Output Specification: Follow the example provided below as closely as possible. In particular, when printing out a polynomial, only print out leading minus signs. Preceding all other positive terms, print a plus sign. For all negative terms, do NOT precede them with a plus sign. Also, if the coefficient of a term is 1, do not print out a one preceding the term. Thus, the polynomial {(2,3), (-1, 1)} should be printed as `2x^3 -x` and not `2x^3 + -x` or `2x^3 -1x`.) Notice that there is one space between each term and the following term or plus sign and one space in between a plus sign and the its following term.

Restrictions: Name the file you create and turn in *poly.c*. Although you may use other compilers, your program must compile and run using gcc or cygwin. Your program should include a header comment with the following information: your name, course number, section number, assignment title, and date. You should also include comments throughout your code, when appropriate.

Deliverables: A single source file named *poly.c* turned in through WebCT.

Sample Output:

```
     Welcome to the CS1 Polynomial Representation System
                      -Select your option-

1. Create a new polynomial.
2. Print an existing polynomial.
3. Differentiate an existing polynomial.
4. End session.

ENTER YOUR SELECTION:  1
How many terms are you entering?
3
Enter the coefficients and exponent pairs.
31 0
41 1
59 2
Polynomial #1 constructed

ENTER YOUR SELECTION:  1
How many terms are you entering?
3
Enter the coefficients and exponent pairs.
27 0
61 4
-9 2
Polynomial #2 constructed

ENTER YOUR SELECTION:  2
Which polynomial do you want to print?  7
Sorry, polynomial #7 does not exist.

ENTER YOUR SELECTION:  2
Which polynomial do you want to print?  2
Polynomial #2 is 61x^4 –9x^2 + 27

ENTER YOUR SELECTION:  3
Which polynomial do you want to differentiate?  2
Polynomial #2 successfully differentiated.
Stored as polynomial #3.

ENTER YOUR SELECTION:  2
Which polynomial do you want to print?  3
Polynomial #3 is 244x^3 –18x

ENTER YOUR SELECTION:  4
Polynomial #1 is 59x^2 + 41x + 31
Polynomial #2 is 61x^4 –9x^2 + 27
Polynomial #3 is 244x^3 –18x
GOOD BYE!
```