Programming Assignment 2
CSCI 235
Ilya Korsunsky
Due Tuesday March 24 (midnight)

Follow the instructions presented in the Programming Rules documents. Submit only the header and source files as well as your Makefile. Note that your program needs to compile in order to get any credit at all.

Adhere to the style guidelines.

Start Early!

Modify the LinkedBag implementation in order to implement operations on **polynomials** of one parameter. For example the polynomial

-3*x^7 + 4.1 * x^5 + 7 * x^3 + 9 * x^0

can be represented as:

head_ptr_→ [-3, 7] → [4.1, 5] → [7, 3] → [9, 0] → nullptr

The degree of the above polynomial is 7.

In order to achieve this you need to do the following:

a) Modify the Node class so that it now holds two items: coefficient_ and exponent_. The first node in the above example has coefficient_ = -3 and exponent_ = 7. That means that instead of GetItem() you will have GetCoefficient() and GetExponent(), instead of SetItem() you will have SetCoefficient() and SetExponent(), etc. Modify all other functions accordingly.

b) You will also need to modify the LinkedBag class in various ways. You can now call it LinkedPolynomial. *Also, there is no need to use the BagInterface.h class.*
   i. Remove the functions ToVector() and Remove() from the LinkedPolynomial implementation.
   ii. Modify the Add() function so that it now adds at the end of the list. Note, that now the Add() will take two parameters, a coefficient and an

exponent. Also Add() should not add a node with an exponent that is already there. So for example the following code:

```
LinkedPolynomial<double> polynomial;
polynomial.Add(-3, 7);
polynomial.Add(4.1, 5);
polynomial.Add(7, 3);
polynomial.Add(8.1, 5);
```

Should result to

head_ptr_→ [-3, 7] → [4.1, 5] → [7, 3] → nullptr

The last Add() did not have any effect because a node with exponent 5 was already there.

iii. Add a member function called DisplayPolynomial() that will traverse the list and will cout the coefficients and exponents. Note that this is a const function. Display in the following form:

-3 * x^ 7 + 4.1 * x^ 5 + 7 * x^3 + 9 * x^0

iv. Add a member function called Degree() that returns the degree of the polynomial (or -1 if the polynomial is empty). Note that this is a const function.
In the previous example polynomial.Degree() should return 7.

v. Add a member function called ItemType Coefficient(const ItemType& exponent) that will return the coefficient of a given exponent. For example polynomial.Coefficient(5) should return 4.1. Note that this is a const function.

vi. Add a member function called bool ChangeCoefficient(ItemType new_coefficient, ItemType exponent) that changes a coefficient for a given exponent. For example if you call polynomial.ChangeCoefficient(100, 3) the resulting polynomial will change to

head_ptr_→ [-3, 7] → [4.1, 5] → [100, 3] → [9, 0] → nullptr

The function returns true if the exponent was in the original polynomial and false otherwise.

In order to test the above do the following:

a) Write a client function (you can place it on top of main()) LinkedPolynomial<double> CreatePolynomialFromInput() that prompts the

user to provide a sequence of coefficients/exponents and then uses the Add() member function to add them to the Polynomial.

b) Then write a client function called TestPolynomial() that first calls CreatePolymomialFromInput() to generate a new polynomial and then does the following in sequence:

Calls the DisplayPolymomial() function.
couts the Degree() of the Polymomial.
Asks the user to provide an exponent.
couts the Coefficient(exponent).
Asks the user for a new coefficient (call it new_coefficient).
Calls the functions ChangeCoefficient(new_coefficient, exponent) and couts its return value (either true of false).
Calls the DisplayPolynomial() function.

Finally, write a member function to **add a given polynomial** to the current one: void AddPolynomial(const LinkedPolynomial<ItemType> &b_polynomial). So if the b_polynomial is

head_ptr_→ [1, 8] → [2, 5] → [-8, 0] → nullptr

Then the current polynomial will change to:

head_ptr_→ [1, 8] → [-3, 7] → [6.1, 5] → [7, 3] → [1, 0] → nullptr

In order to test the above write a function called TestAddition() that does the following:

Calls the function CreatePolynomialFromInput() twice to generate two polynomials polynomial1 and polynomial2. Adds the second one the the first one (i.e. calls polynomial1.AddPolynomials(polynomial2)). And finally displays the result by calling polynomial1.DisplayPolynomial().