# Department of Computing

**CS250: Data Structure and Algorithms**

**Class: BSCS-6C**

**Lab 6: Linked List Functionality**

**Date: October 20, 2017**

**Time: 0900 to 1200**

# Instructor: Abid Rauf

# 

**Lab 6: Linked Lists Functionality**

**Introduction**

In this lab, you will use your knowledge of linked lists to implement a small functionality.

**Objectives**

This lab will introduce students with the practical implementation of linked list with its operations.

**Tools/Software Requirement**

Visual Studio C++

**Description**

**Representing a polynomial using a linked list**

A polynomial can be represented in an array or a linked list by simply storing the coefficient and exponent of each term. However, for any polynomial operation, such as addition or multiplication of polynomials, you will find that the linked list representation is easier to deal with. First of all note that in a polynomial all the terms may not be present, especially if it is going to be a very high order polynomial. Consider,



Now this 12th order polynomial does not have all the 13 terms (including the constant term). It would be very easy to represent the polynomial using a linked list structure, where each node can hold information pertaining to a single term of the polynomial.

Each node will need to store,

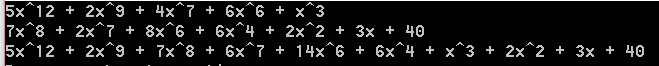
- The exponent, and

- The coefficient for each term.

Thus, we need to define a node class to hold two integers (i) exponent and (ii) coefficient.



**Figure 1: Linked List**



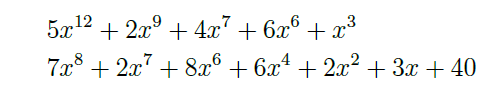
**Figure 2:Output of the program**

**Lab Task:**

Addition and multiplication of two polynomials. Consider addition of the following polynomials,

**Assumption:**

1. The polynomials will only be using addition operation
2. Both the expressions are ordered with respect to their exponents as shown below.



The resulting polynomial is going to be,



Now notice how the addition was carried out. (Assumption: One object of a Polynomial class will be called with its operation of addition with the second Polynomial as a parameter.)

* We start with a check to see if either of the lists are empty.
* Then we start an iterative process of adding the nodes to the list as following:
  + If the current nodes of both the lists have equal exponents then we just add the coefficients of the two and move our pointer to the next node and continue the process.
  + If current node’s exponent is greater than the current node of the list passed as a parameter then we simply move our pointer to this list to its next node and continue the process.
  + If current node’s exponent is smaller than the current node of the list passed as a parameter then we simply remove that node from the list passed as a parameter and insert it to its appropriate position in the list and continue the process.
  + At the end we simply check if we have some more nodes in the list passed as a parameter and insert them to their appropriate locations.
* Execute the completed code, and the output should be something similar one in Figure 2.
* Some important checks are the following:
  + One should not be able to remove a node when the list is empty and an error message (ERROR: The list is empty) should be displayed when such an attempt is detected.
  + ERROR: Zero coefficient is not allowed should be displayed when someone attempts to insert a node with zero coefficient.
  + ERROR: Greater or equal exponents are not allowed should be displayed if someone attempts to insert a node with exponents that is equal to or greater than the last element’s exponent in the list.
  + ERROR: Null polynomials are not allowed should be displayed if someone attempts to add two lists such that one of them is null.

**Minimum requirement in the main function is the following:**

Polynomials p1, p2;

//Check to see if we can remove a node here.

p1.append(5, 12);

p1.append(2, 9);

//Check to see if we can insert a node with zero coefficient here.

//Check to see if we can insert a node with an exponent which is already there in the list

//Check to see if we can insert a node with an exponent which is greater than the last element in the list

p1.append(4, 7);

p1.append(6, 6);

p1.append(1, 3);

p1.display();

p2.append(7, 8);

p2.append(2, 7);

p2.append(8, 6);

p2.append(6, 4);

p2.append(2, 2);

p2.append(3, 1);

p2.append(40, 0);

p2.display();

p1.add(p2);

p1.display();

system("pause");

**Deliverable**

Students are required to upload the lab task on LMS before the deadline. Compile a single Word document by filling in the solution/answer part and submit this Word file on LMS.

This lab is graded. Min marks: 0. Max marks: 10.