Name ___________________________________

CS 235 Midterm 1
Version 1.3
Instructors: K. Seamons and R. P. Burton

Oct 18-22 (Thursday through Monday)
Due in the Lab on Monday not later than 7:00 p.m.
Penalty for submitting the midterm late:
20 points per day (including the weekend), advancing at 7:01 p.m. each day

Open Book (142 course text and your CS 235 course text only), Open Notes (including your own Lab solutions)
Open Secondary Storage Device: yours only
Open Laptop: if you wish
Open Course Website and C++ API, but no other Internet resources (such as non API resources on the C++ website)
Closed Neighbor (and everyone is thy neighbor)

Instructions
(Please read carefully)

1. This midterm consists of a C++ programming problem with optional extra credit. Read and understand the statement of the problem completely before beginning to design, code, and test. As part of your design, consider the test cases that will establish the correctness of your solution. Test your solution thoroughly before submitting it.
2. Produce a solution, which consists of your C++ code, with a comment at the beginning of each file (both .h and.cpp) which includes your name, your student ID number, and "CS 235 Fall 2012 Midterm 1." Upload your completed project by compressing the files and submitting through Gradebook with TA assistance. If a packet is not collected by a TA upon submission, you will not be graded and will therefore receive no credit on the exam. Attribute any code taken from or based on other sources (except for the course texts and the course websites). Attributed code copied from or based heavily on outside sources is worth half credit. Unattributed code copied from or based heavily on outside sources is worth no credit.
3. Understanding the problem correctly is part of the examination. If something seems unclear, ask a CS 235 TA for clarification. You may pose questions to the CS 235 TAs at any time. However, the TAs generally are not permitted to answer questions related to design, C++ implementation, debugging, or testing.
4. Prior to submitting your midterm, score it using the attached scoring sheet (this will help you maximize your points and will help us grade your exam accurately).
5. When you are finished, go to the course website and follow the link labeled "Submit Exam" in the Exam Menu.
6. Sign here to request that your midterm be graded and to certify that no unfair information related to the midterm has been received by you, either directly or indirectly, and that none will be conveyed by you. If we discover that you cheated or assisted someone in cheating, intentionally or unintentionally (including accidentally), your score for this exam may (and probably will) be rand() % 1.

   We're serious.

______________________________________ ________________, 2012
(Signature) (Date)

Requirements

- Extend the `PolynomialManagerInterface.h` and the `PolynomialList.h` classes. Provide working classes that implement all of the method headers given in these files.
- Create linked structures that will store an ordered polynomial. You must implement your own list class. You may not use any predefined data structures from the STL (including Arrays).
- Add new terms to your polynomial lists upon request and in the proper order.
- Provide functionality to add and subtract one polynomial from another resulting in a third polynomial stored in an *ordered* list.
- The list should be ordered by descending exponent with the highest exponent found in the node at the beginning of the list.
- Reject invalid input by following the instructions given in the source files.
- Print a list in the correct ordered fashion as shown in the examples.

Clarifications & Constraints

- For purposes of this midterm, a term is valid if
  - it consists of a variable, a '^', and an exponent. There may be a coefficient as well; however in the absence of a coefficient, a 1 is implied and expected
  - exponents are non-negative integers; however in the case of a 0 in the exponent, the variable will not be printed, and will be treated as an implied 1. If a 1 appears as the exponent, only the x should be printed (see example below)
  - the variable is an 'x'
  - it has a space between each component, e.g. "12 x ^ 45"
- Your program needs to handle all exceptions. This requires you as the programmer to anticipate what problems could arise from invalid input and reject them accordingly.
- There will be a Test Driver used to grade the exam; however it will not be available for student use. You should test your code thoroughly before submission.
- When printing, the links between your Nodes should be represented with an addition sign ('+').

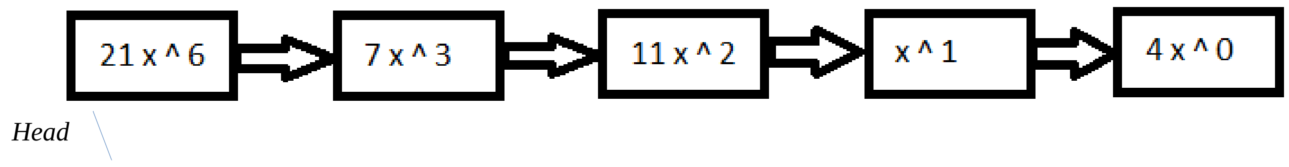
Extra Credit (precondition: submission of all required parts of the midterm)

- (5 points) Memory Management. You have properly handled memory management, and there are no memory leaks at the end of the program.

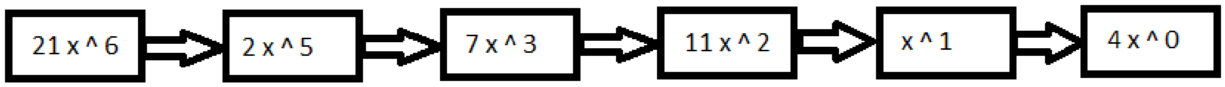
Examples

- If given the terms "21 x ^ 6", "7 x ^ 3", "11 x ^ 2", "x ^ 1", and "4 x ^ 0"

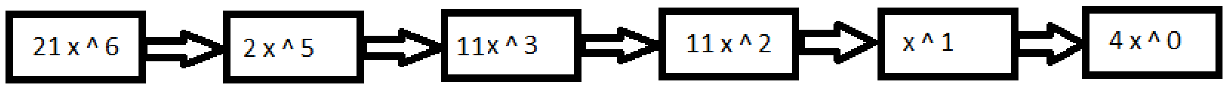
Your list should look like this:



- When inserting the new term “2 x ^ 5” into the list, the new list should look like this:



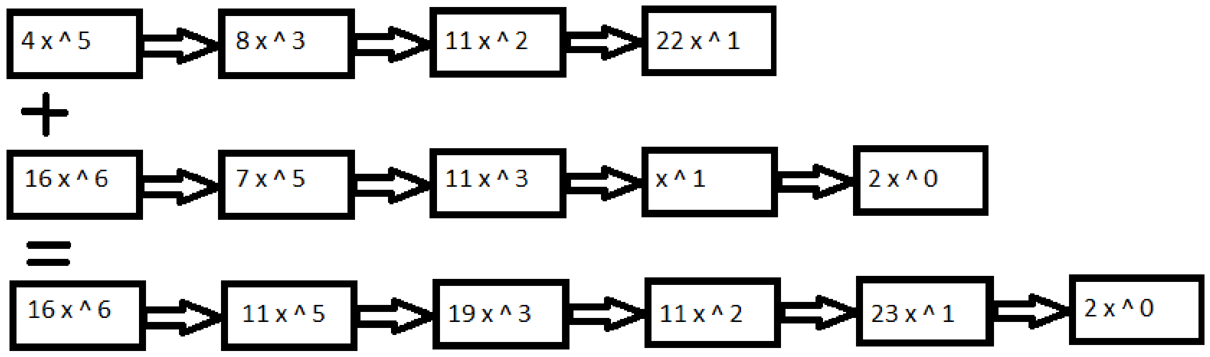
- When inserting the new term “4 x ^ 3” into the list, the new list should look like this:



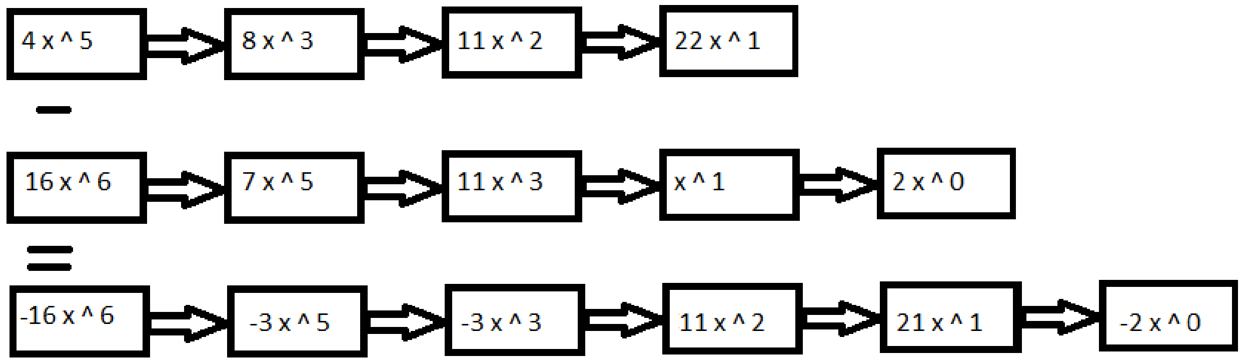
The list printed should return a string that looks list this:

“21 x ^ 6 + 2 x ^ 5 + 11 x ^ 3 + 11 x ^ 2 + x + 4”

- If given two linked lists and asked to add the lists together, the resulting list should be:



- Given two linked lists and asked to subtract the lists from one another, the resulting list should be as follows (Note. This is the only situation in which you would end up with a negative coefficient, e.g. in the resulting returned list):



Printing this Result List should look like this:
“-16 x ^ 6 + -3 x ^ 5 + -3 x ^ 3 + 11 x ^ 2 + 21 x + -2”

Midterm 1 Scoring Sheet

PIN:___________________BYU ID #:_______________ Day Submitted: ___________ T.A. ___
Days Late: _____

Student Grading	TA Grading
___/ 25 pts	___/ 25 pts – Using Linked Lists
___/ 10 pts	___/ 10 pts – Inserting into lists correctly
___/ 5 pts	___/ 5 pts – Clearing lists correctly
___/ 5 pts	___/ 5 pts – Using nodes properly
___/ 5 pts	___/ 5 pts – Storing the correct data in the nodes
___/ 25 pts	___/ 25 pts – Required Functionality
___/ 15 pts	___/ 15 pts – Adding terms into both lists
___/ 5 pts	___/ 5 pts – Ordering correctly
___/ 5 pts	___/ 5 pts – Combining like terms
___/ 40 pts	___/ 40 pts – Combining the Lists
___/ 20 pts	___/ 20 pts – Adding lists together
___/ 20 pts	___/ 20 pts – Subtracting lists from one another
___/ 10 pts	___/ 10 pts – Code Style
___/ 5 pts	___/ 5 pts – Neat Code: correct indentation, comments as needed, helpful variable names
___/ 5 pts	___/ 5 pts – No debugging `cout` statements or any print/pause statements in submitted files
___/ 5 pts	___/ 5 pts – **Extra Credit**
___/ 5 pts	___/ 5 pts – Memory Management
___/ 100 pts	___/ 100 pts – Total (before late penalties)

Student to TA
Comments:____

____

____

____

____

TA to Student
Comments:____

____

____

____

____