

1 Polynomial Representation

1.1 Summary

You will design and implement a Data Structure for Polynomial Representation, Arithmetic, and Evaluation. The implementation should allow for basic arithmetic operations and evaluation. The details of the design of the structure is largely at your discretion. Major factors to consider for the design should be theoretical efficiency (both time and space), practical efficiency, and appropriate memory management. Although some tips are provided below (and have been provided in class), the determination of *how* to make the structure efficient is largely your charge.

1.2 Programming Languages

I encourage you to submit your projects using C++. You may choose a different programming language with prior approval from me. However there are caveats as not all programming languages have the same characteristics. Also note, if choosing a language other than C++, you may by chance choose a language with which the TAs are not familiar, thus limiting the amount of assistance they might provide. Note one of the main goals of our class projects is for you to learn how to construct various data structures from the most *elemental programming constructs*. Thus you will not receive credit when using any pre-existing structures from programming libraries or code that has been created or designed by others. For example in C++ you cannot use the pre-existing vectors, stacks, lists, etc.

Note: There are many versions of C++. You must use the version that is currently running on the course server.

Please note that complex data structures (non-elemental constructs) are “built-into” some programming languages. If this is the case, you cannot use the built-in structure. For example, in Python, both list and “array” structures are fairly complex and not elemental programming constructs, e.g., they can change size dynamically. If you are using Python, you will not receive credit when using these structures.

If you have any questions as to what structures are permitted (and which are not permitted), given your language of choice, please ask me.

1.3 Planning and Design

Before implementation, you should plan and design using standard approaches, e.g. UML class diagrams, flow diagrams, etc. If you have questions pertaining to your project, I will

first ask to see your designs. **I will not look at your code without first viewing your design documents.**

You will be faced with many design decisions during this project. It is best to spend the requisite time during the design stages to assure an appropriate and efficient implementation is built. Consider your options, perform a theoretical complexity analysis of the different options, and base your decision on the results of your analysis.

1.4 Input Requirements

The program will take one command line argument, which will be an input file. The input file will contain the information needed to construct 2 polynomials. The program will load the polynomial information from the input file. The first line will contain pairs of numbers to represent a term in the polynomial, given the standard polynomial expansion. The first number will be the coefficient and the second number will be the exponent. For example, if the first line is 1 4 3 2 -1 0, this should be interpreted as $x^4 + 3x^2 - 1$. You may assume all the exponents in the terms of each polynomial will be non-negative integers. You may assume all coefficients are integers. A sample input file will be provided.

1.5 Structural and Operational Requirements

1. provide efficient polynomial representation
 - Efficiently stores polynomial information
 - Appropriately displays to screen polynomial information, e.g. $6x^7 - x^5 + 3x^2 + 2$
 - Appropriately allocate / deallocate memory
 - Valid state is always maintained
2. efficiently performs polynomial addition (overload +)
 - adds 2 polynomials (resulting in a new polynomial), e.g. assume p and q are polynomials: $p = (6x^7 - x^5)$ and $q = (33x^7 + 2x^4 - 1)$. Then $p + q = 39x^7 - x^5 + 2x^4 - 1$
3. efficiently performs polynomial multiplication
 - multiplies 2 polynomial operands (resulting in a new polynomial), e.g. $(6x^7 - x^5) * (x^7 + 2x^4 - 1) = 6x^{14} - x^{12} + 12x^{11} - 10x^9 - 6x^7 + x^5$
4. efficiently performs polynomial exponentiation
 - raises polynomial operands to integer power (resulting in a new polynomial), e.g. $(x^3 + 3x)^n$, where n is a non-negative integer. Example: $(x^3 + 3x)^2 = x^6 + 6x^4 + 9x^2$
5. efficiently performs polynomial evaluation

- evaluates a polynomial given the provided parameter value, for example, if $p = (6x^3 - x^2)$, p evaluated at $x = 2$ should result in value 44.

6. efficiently checks for overflow errors

- if an overflow is detected at some stage of computation, then the final result will be invalid. Set the resulting value to a sentinel. Efficiently proceed to the next computational task. *Hint: Make use of throwing an exception, and then catch it appropriately.*
- Design your program to minimize the chance of an overflow error.

Notes: To earn full marks it is expected that you will make the polynomial structure very efficient (in space and time). If you are faced with a space / time tradeoff, you will opt to improve time complexity (if the cost of space is relatively minor). For example, you may wish to use previously (in-class) discussed mathematical methods and computational operators that provide for fast implementation.

1.6 Output Requirements

The main method should read in the input file and display the resulting polynomials, and their specified: addition, multiplication, subtraction, exponentiation and evaluation as denoted below. (The main method should also be efficient.) If an overflow is detected for any one of the requirements below, print to the screen “Overflow Detected” for that corresponding line of output.

For each input file, the following output is expected (enumerated in this order):

1. Print out standard polynomial representation for the first polynomial read in from the input file. AND Print the value of this polynomial for $x = 10$.
2. Print out standard polynomial representation the second polynomial read in from the input file. AND Print the value of this polynomial for $x = 10$.
3. Print out standard polynomial representation for the resulting polynomial of the sum of the two input polynomials. AND Print the value of the sum for $x = 10$.
4. Print out standard polynomial representation for the resulting polynomial of the product of the two input polynomials. AND Print the value of the product for $x = 2$.
5. Assume the first input polynomial is p_1 and the second input polynomial is p_2 . Print out standard polynomial representation for the following 2 resulting polynomials and 1 resulting value:
 - p_1^3
 - $p_1^{p_2.eval(5)}$

No other outputs should be observed.

1.7 Submission and Compilation Requirements

Please provide extensive comments prior to each method and class. Justify your design decisions within your comments. Submission Deadline – See Canvas. Budget your time well. Include significant time for design / planning and testing / debugging. Please submit early and often! Your last submission will be graded.

Your code must run on the class server.

To standardize submissions, you will submit a makefile, which will contain the necessary compilation commands for your code. The target executable will be named p1.

Thus, the following steps should run your code on the course server (CHECK IT).

```
make p2
./p2 inputFilename
```

If the program does not compile (following your instructions) or the program does not run, the submission will not be accepted.

1.8 Testing and Debugging (Optional)

You may wish to construct an interactive interface to test the functionality of your structure at intermediate stages of development. This would likely be most efficient with an interactive interface that allowed you to interactively test various functionalities of your structure given different inputs. *If you do implement a testing interface, please be sure to comment it (so that it does NOT execute) before submission.*

1.9 Rubric

List of Requirements	Percentage
efficient infrastructure for polynomial representation and display	0.40
efficient polynomial addition	0.10
efficient polynomial subtraction	0.10
efficient polynomial multiplication	0.10
efficient polynomial evaluation	0.10
efficient polynomial exponentiation	0.10
efficient main	0.05
efficient overflow error detection and “recovery” (program continues)	0.05
TOTAL	1.0

Each section of the Rubric List will be evaluated based on correctness and efficiency. About 1/2 of the points for each item in the requirements list are awarded on correctness, and about 1/2 of the points for each item in the requirements list are awarded on efficiency. Thus you will need to perform a theoretical analysis of complexity before deciding on an implementation strategy. It may also be a good idea to check the practical efficiency of your code using the unix “time” command; however please note that your code will be graded based on the code written and not the true runtime of the algorithm. During the design and coding process, you will likely make many design modifications which should result in improved efficiency. Note, c compilers will optimize your code during compilation ,and thus you may not see any efficiency gains (e.g. reduced runtime) when explicitly making the design modifications in your code. This is OK and should be expected for some modifications. HOWEVER, please leave these design elements in your code. (It is best not to always rely on the compiler as not all compilers are the same.)