Homework #1

CSCI2421 – Summer 2021 – J. Pastorino

Due: *Check Canvas*

1 Description

This homework will test your understanding of the topics from Chapters 1 through 4 and your ability to implement a simple ADT using the learned tools.

This homework is to be completed alone and without help. Do not share your code, or part thereof, over public channels, such as Slack. Copying code from the internet and submitting it as your own is considered plagiarism. A grade of 0 will be set in those cases, and other measures may be taken in those situations.

2 Problem Description

You are asked to implement an ADT `SparsePolynomial`, to represent simple polynomials such as $-3x^7 + 4x^5 + 7x^3 - 1x^2 + 9$.

The design of the ADT is already completed for you and is provided in figure 1. Section 3, described the methods with more detail.

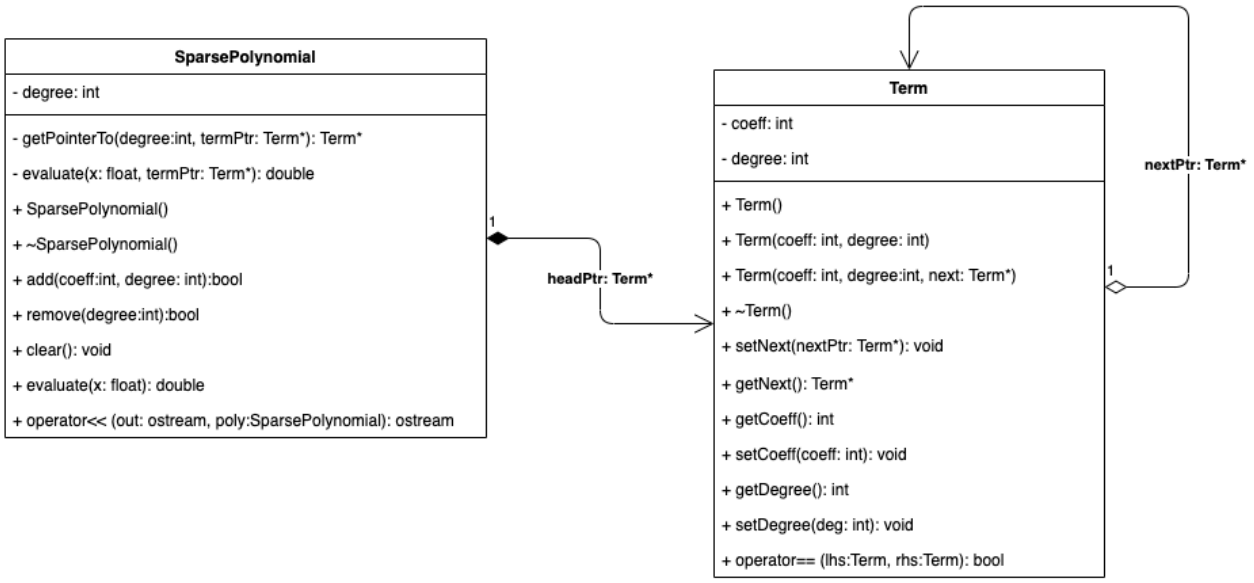
SparsePolynomial
- degree: int
- getPointerTo(degree:int, termPtr: Term*): Term*
- evaluate(x: float, termPtr: Term*): double
+ SparsePolynomial()
+ ~SparsePolynomial()
+ add(coeff:int, degree: int):bool
+ remove(degree:int):bool
+ clear(): void
+ evaluate(x: float): double
+ operator<< (out: ostream, poly:SparsePolynomial): ostream
1
headPtr: Term*
Term
- coeff: int
- degree: int
+ Term()
+ Term(coeff: int, degree: int)
+ Term(coeff: int, degree:int, next: Term*)
+ ~Term()
+ setNext(nextPtr: Term*): void
+ getNext(): Term*
+ getCoeff(): int
+ setCoeff(coeff: int): void
+ getDegree(): int
+ setDegree(deg: int): void
+ operator== (lhs:Term, rhs:Term): bool
nextPtr: Term*
1

Figure 1: UML Class Diagram for ADT SparsePolynomial

The class `SparsePolynomial` represents the polynomial to which you can add and remove terms, and you can evaluate the polynomial for a particular value such as $f(10)$. The class `Term` represents each term of the polynomial, such as $4x^5$ from the aforementioned example.

3 ADT Description

3.1 SparsePolynomial

This class represents the sparse polynomial. The idea of *sparse* comes as only those terms with coefficients greater than zero will be represented in the data structure.

- `add`: takes a term's coefficient and degree and will add a term with that coefficient for the given degree. E.g. `add(3,5)` adds the term $3x^5$ to the sparse polynomial. If the polynomial, has already a term for that degree, then the coefficient will be updated to the newly provided coefficient. `coeff` can be any integer, and `degree` is an integer ≥ 0.
- `remove`: removes the term with that degree.
- `clear`: removes all the terms in the polynomial.
- `evaluate(x)`: this **must be a recursively implemented**, and it returns the value of evaluating the polynomial with the given x value. E.g. if the argument is 5 then the result will be
- `evaluate(x, termPtr)`: is the recursive implementation used by `evaluate(x)`. See `LinkedBag` for reference on implementing recursive methods. E.g. the result of evaluating $-3(5)^7 + 4(5)^5 + 7(5)^3 - 1(5)^2 + 9$ will be the result returned for the polynomial presented above.
- `operator<<`: will write in the output stream the polynomial represented as an equation. E.g. for the polynomial represented in the figure 2, the output will be: `_3x^7_+_4x^5_+_7x^3_+_(-1)x^2_+_9_`. The "_" represents a single space. No new line should be added to the output.
- `getPointerTo` is analogous to the one described for the `LinkedBag`. This function uses recursion.

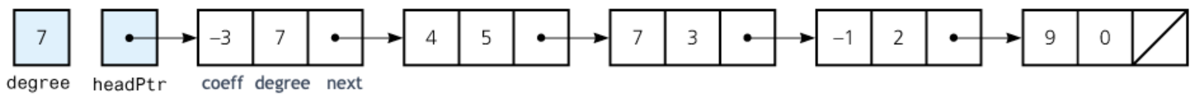

Figure 2: An internal representation of the space polynomial $-3x^7 + 4x^5 + 7x^3 - 1x^2 + 9$.

3.2 Term

This class represents the terms of the polynomial.

- A term $5x^{10}$ will be represented as an object `Term` with $coeff = 5$ and $degree = 10$.

- `operator ==`: returns whether the left hand side term and the right hand side of the term are the same. Two terms are the same if both the `coeff` and the `degree` in both terms are the same.

4 ToDo

1. Implement the `SparsePolynomial` ADT.
2. Document your code following the given guidelines.
3. Test your program properly. A short demo program (`main.cpp`) is provided for you, however this is not comprehensive.

5 Restrictions

This restrictions apply for this homework:

- You can only include the following libraries:
  - `cmath`
  - `cstddef`
  - `iostream`
  - `string`
- You can use `pow` to evaluate the terms.
- `SparsePolynomial::evaluate` **must be a recursively implemented**

6 Submission and Grading

This homework will be automatically graded, meaning that you will have access to an automated system (*autograder*) where you will submit your source files, and a program will check your submission. This program will output compilation errors if your program does not compile, or the number of tests that your program succeeded, providing as much information as possible for your to review your program.

This *autograder* will be available for your to access through a webpage and requires you to be connected to VPN. Once you submit your code, you will receive a notification through your your UCD email, this may take between a couple minutes and no more than 15 minutes, depending on the server load. If you don't get the notification within that time range, contact your TA or instructor. Then, through the website you can access your results.

You can submit your code to review as many times as you needed.

Once you are satisfied with your solution, you will submit the same *zip* file you use to submit to *autograder* as your **final submission to Canvas**, which will then be reviewed by the TA and finally graded.

There are some restrictions and requirements that might require a revision by your instructors. Make sure you follow those otherwise you will not receive marks, even if your automatic solution is correct. E.g. if you are required to submit a recursive implementation and your code is iterative, you will not receive grade.