CPSC 2430 Data Structures

Homework Assignments #1 (Warm-up)

1. Problem

You need to design and implement a polynomial class named `Poly` that represents a polynomial of a single variable x with degree n: $a_0 + a_1x + a_2x^2 + \ldots + a_nx^n$, where $a_0, a_1, \ldots, a_n$ are integer constants called the **coefficients** of the polynomial. In this assignment, we assume $0 \le n \le 100$. Please refer to https://en.wikipedia.org/wiki/Polynomial for more details on polynomials.

For example, $1 + 3x - 7x^3 + 5x^4$ is a polynomial of degree 4 with integer coefficients 1, 3, 0, -7 and 5, in order of their degrees (0, 1, 2, 3, and 4). A common implementation of a polynomial stores the degree of the polynomial and the list of coefficients.

The `Poly` class must support the following operations as shown in the table below. **You are forbidden to change the class name and function prototypes**.

`Poly()`	Default constructor: construct an empty polynomial (e.g., its degree = 0).
`void read()`	Read the polynomial's degree and coefficients from `cin`. For example, if we want to input a polynomial $1 + 3x - 7x^3 + 5x^4$, it shall first read its degree 4. Then, it shall read its five coefficients 1, 3, 0, -7, and 5 in order. In other words, the first non-negative number shall be degree, followed by a list of coefficients. Given a list of integer inputs 4, 1, 3, 0, 7, 5, the polynomial is $1 + 3x - 7x^3 + 5x^4$. Input validation for the polynomial degree should be performed.
`void display() const`	Write the polynomial to `cout`. For a polynomial $1 + 3x - 7x^3 + 5x^4$, it shall be displayed in the form of 1+3x–7x^3+5x^4. The terms with a coefficient of zero shall not be displayed.
`Poly operator+(const Poly& p)`	Overloading operator +. Add two polynomials.
`Poly operator-(const Poly& p)`	Overloading operator -. Subtract one polynomial from the other.
`int evaluate(int v)`	Return the value of the polynomial for the variable x=v.

	A large value of v can cause integer overflow when evaluating the polynomial, especially when the degree n is large. Instead, we return the value of the polynomial modulo the prime number 1,000,000,007.
	
	For Modular Arithmetic, please refer to this link for more details: https://www.geeksforgeeks.org/engineering-mathematics/modular-arithmetic/
	
	Note: When two integers a and b are multiplied, the result may cause integer overflow. A simple way to avoid this is as follows.
	
	`const int MOD = 1000000007;`
	`int c = (long long) a * b % MOD;`
	
	`or:`
	
	`int c = static_cast<long long>(a) * b % MOD;`

Question: what data structure can be used to store the coefficients of a polynomial? Alternatively, how do you represent a polynomial in memory?

Before moving to the answer on next page, please try to answer it by yourself first.

Answer: You can use a static array, dynamic array or linked list. But if you use a dynamic array or linked list, your Poly class must implement copy constructor, assignment operator and destructor.

For simplicity, you can use a **static array** in this assignment.

2. The Client (or Driver) Program

A `client.cpp` will be provided for this assignment. However, feel free to implement your own `client.cpp` during testing and debugging. Please note that the provided `client.cpp` needs to be included in your final submission.

Specifically, the client program `client.cpp` needs to perform operations similar to what follows.

(1) Test default constructor, read(), and display().

```
Poly p1, p2;
p1.read();
p1.display();
p2.read();
p2.display();
```

(2) Test overloading operators + and -.

```
Poly p3;
P3 = p1 + p2;
p3.display();
Poly p4;
P4 = p1 – p2;
p4.display();
```

(3) Test evaluate(). E.g.,

```
cout << p1.evaluate(55) << endl;
```

A snapshot of my program's execution is shown below for your reference. Your `client.cpp` implementation does not have to match mine. My implementation demonstrates only the minimum required functionality and does not optimize the user interface.

```
[zhuy@cs1 HW1]$ ./client
Degree of the polynomial: 4
Input the polynomial coefficients in increasing order their degrees:
1 -3 4 5 6
The inputed polynomial is: 1-3x+4x^2+5x^3+6x^4

Degree of the polynomial: 6
Input the polynomial coefficients in increasing order their degrees:
1 3 -4 -5 6 7 8
The inputed polynomial is: 1+3x-4x^2-5x^3+6x^4+7x^5+8x^6

Adding two polynomial.
1: 1-3x+4x^2+5x^3+6x^4
2: 1+3x-4x^2-5x^3+6x^4+7x^5+8x^6
The result: 2+12x^4+7x^5+8x^6

Subtracting two polynomial.
1: 1-3x+4x^2+5x^3+6x^4
2: 1+3x-4x^2-5x^3+6x^4+7x^5+8x^6
The result: -6x+8x^2+10x^3-7x^5-8x^6

Evaluation: input x value for p3: -6x+8x^2+10x^3-7x^5-8x^6
189
The result is: 870919075
[zhuy@cs1 HW1]$
```

3. Submission

You need to submit the following files:

- poly.h: header file for Poly class. You need to include Redundant Declaration in the header file.
- poly.cpp: implementation file for Poly class.
- client.cpp: the provided client program.
- Makefile (provided).

Before submission, you should ensure your program has been compiled and tested (extensively) in cs1.seattleu.edu. Your assignment receives zero if your code cannot be compiled and executed.

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, you should follow two steps below (assuming your files are on cs1.seattleu.edu):

- Pack all your files into a package named ***hw1.tar***

```
tar -cvf hw1.tar poly.h poly.cpp client.cpp Makefile
```

- Submit the package ***hw1.tar*** as the first programming assignment ***HW1***

```
/home/fac/zhuy/class/submit2430 HW1 hw1.tar
```

The message similar to the following one will be displayed if the submission is successful.

```
======================Copyright(C)Yingwu Zhu=========================
Sun Mar 28 23:01:52 PDT 2021
Welcome testzhuy!
You are submitting hw1.tar for assignment HW1.
Transferring file.....
Congrats! You have successfully submitted your assignment! Thank you!
Email: zhuy@seattleu.edu

======================================================================
```

4. Grading Criteria

Label	**Notes**
[1] Submission (1 pt)	All required files are submitted.
[2] Makefile (1 pt)	Makefile compiles the code and generates the executable.
[3] Presentation (1 pt)	Clean, well-commented code. No unsolicited output messages such as testing & debugging messages.
[4] Functionality (6 pts)	The Poly class should be implemented as specified: all six member functions are properly implemented. All six member functions are tested in client.cpp (client.cpp is well written to cover all the required test cases.). The read() member function should validate the degree input.
[5] Overriding policy	If the code cannot be compiled or executed (segmentation faults instantly, for instance), it results in zero point for Functionality. No further investigation will be conducted on your program.
[6] Late submission	Please refer to the Syllabus for details.