CSE2122 (Spring 2020)
Homework 3, due 2/14/2020 11:59pm

Before you start, please login your stdlinux account and download the solution file and the template program from the class directory to your stdlinux directory. You can use the command **"cp /class/cse2122/hw3/* ./."**.

Submit your code to the designated dropbox on Carmen under the Assignments tab. Late submissions receive a 10% penalty for each day. No late submissions after the second day.

Skills needed to complete this assignment: dynamic arrays, classes.

In mathematics, a polynomial is an expression consisting of variables and coefficients, which involves only the operations of addition, subtraction, multiplication, and non-negative integer exponents of variables. A polynomial in a single variable can always be written in the form

$$a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$

where $a_0, \ldots, a_n$ are coefficients and x is the variable.

In this assignment, you will complete a polynomial class using dynamic arrays. For simplicity, we only consider polynomials with one variable and integer coefficients.

One simple way to implement the polynomial class is to use an array of integers to store the coefficients. The index of the array is the exponent of the corresponding term. If a term is missing, then it simply has a zero coefficient. For example, we can use [2, 0, 0, 4] to represent the polynomial $4x^3 + 2$, since the polynomial can be rewritten as $\mathbf{2}x^0 + \mathbf{0}x^1 + \mathbf{0}x^2 + \mathbf{4}x^3$.

The `Poly` class provided in the template has two data members, an integer called `arraySize`, and an integer pointer called `coeff`. `coeff` will be used to link the dynamic coefficient array, and `arraySize` will be used to hold the current size of `coeff` array. With these data members, you will complete the following class member functions and friend functions:

- `Poly()`
  The default class constructor. A class member function. It is used to allocate an array of size `DEFAULTPOLY` (a global constant provided in the template) and initialize the polynomial to the constant 0.

- `Poly(int size)`
  An alternate class constructor. A class member function. It is used to allocate an array of size `size` (the input parameter) and initialize the polynomial to the constant 0.

- `Poly(const Poly& aPoly)`
  The copy constructor. A class member function. It is used to construct a new `Poly` that is a copy of an existing polynomial object, `aPoly`. Note that the new `Poly` object does not share the same dynamic array with `aPoly`. A new array needs to be allocated to it.

- `~Poly()`
  The destructor. A class member function. It is used to destroy a `Poly` object by freeing its dynamically allocated array.

- `void grow(int newSize)`
  A class member function. If `newSize` is greater than the current array size, this function will increase the size of the dynamically allocated array by allocating a new array of the desired size, copying the data from the old array to the new array, and then releasing the old array. If `newSize` is less than or equal to the current size, no actions are taken.

- `int degree() const`
  A class member function. The degree of a single-variable polynomial is the highest exponent of its terms with non-zero coefficients. For example, the polynomial $4x^3 + 2$ has degree 3, and the polynomial 2 has degree 0. This function returns the degree of the polynomial.

- `void setCoeff(int value, int i)`
  A class member function. This function is used to add a term $value \times x^i$, in the polynomial. Grow the coefficient array if necessary.

- `int getCoeff(int i) const`
  A class member function. This function is used to find the coefficient of the x^i term in the polynomial. If the polynomial does not contain a term with power i (for example, i >=`arraySize`), the function returns zero.

- `void negate()`
  A class member function. The polynomial will be changed to represent its multiplication by -1 after this function is called. For example, the polynomial $4x^3 + 2$ will be changed to $-4x^3 - 2$.

- `Poly operator+(const Poly& aPoly, const Poly& bPoly)`
  A friend function of the `Poly` class. This function overloads the operator +. This function adds two polynomials and returns a new polynomial that is the result. For example, with $poly1 = 4x^3 + 2$ and $poly2 = 5x^3 + 2x + 2$, $poly1 + poly2$ is $9x^3 + 2x + 4$.

- `Poly operator-(const Poly& aPoly, const Poly& bPoly)`
  A friend function of the `Poly` class. This function overloads the operator -. This function subtracts the second parameter `Poly` object from the first parameter Poly object and returns a new polynomial that is the result. For example, with $poly1 = 4x^3 + 2$ and $poly2 = 5x^3 + 2x + 2$, $poly1 - poly2$ is $-x^3 - 2x$.
  (Hint: Can you call other functions to get the result? Note that $poly1 - poly2 = poly1 + (-1) \times poly2$)

- `bool operator==(const Poly& aPoly, const Poly& bPoly)`
  A friend function of the `Poly` class. This function overloads the operator ==. It is used to decide whether two `Poly` objects represent the same polynomial. Note that for two `Poly` objects representing the same polynomial, their `coeff` array sizes can be different. For

example, coefficient arrays [2, 0, 0, 4] and [2, 0, 0, 4, 0, 0] both represent the polynomial $4x^3 + 2$.

- `ostream& operator<<(ostream& out, const Poly &aPoly)`
  A friend function of the `Poly` class. This function overloads the operator <<. It prints a `Poly` object to the given output stream in a readable format. For example, the polynomial $4x^3 - 2x + 2$ will be printed as `4x^3-2x^1+2`. Note that we will not print the variable nor exponent for a term with zero exponent, e.g. $2x^0$ is printed as `2`. If a polynomial has no nonzero coefficient, it is printed as 0.

Please do NOT modify the class header.

Test cases

Several test cases are provided in the `main` function. For those test cases, you can find the expected outputs by running the solution file. Please note that your grade is decided by the correctness of your `Poly` class, not the results of those test cases.

Program Submission

Compile your code with the following command. Submissions do not compile with the following command will receive zeros.

```
g++ <your file name> -std=c++11 –pedantic-errors
```

Make sure your code is formatted properly. Up to 5% of the assignment credit can be deducted if your program does not have appropriate indention and necessary comments. See textbook on section 2.5 or any code example in textbook/lecture slides.

Submit your code (the .cpp file) to the designated dropbox on Carmen. You can find the dropbox under the Assignments tab.