CS202

Assignment 6: Polynomials

Description

Ah yes, polynomials. In this assignment, you will need to implement a series of operators to perform various calculations on polynomials. A polynomial is essentially a set of coefficients assigned to a term. For example, the following polynomial

$$5x^4 - 3x^2 + 10x + 20$$

can be stored into an array in the following way

20	10	-3	0	5
[0]	[1]	[2]	[3]	[4]

Now that we fully grasp how a polynomial can be stored into memory, let's take a look at the header file

```
class polynomial
{
public:
  polynomial();
  polynomial(const polynomial&);
  polynomial(int*, int);
  polynomial(int);
  ~polynomial();

  polynomial operator*(const polynomial&) const;
  polynomial operator*(int) const;
  polynomial operator+(const polynomial&) const;
  polynomial operator+(int) const;
  const polynomial& operator=(const polynomial&);
  const polynomial& operator=(int);
  polynomial operator-() const;
  polynomial operator-(const polynomial&) const;
  polynomial operator-(int) const;

  friend std::ostream& operator<<(std::ostream& outfile, const polynomial&);
  friend polynomial operator+(int, const polynomial&);
  friend polynomial operator*(int, const polynomial&);
```

```
  friend polynomial operator-(int, const polynomial&);
private:
  int * polyExpr;
  int degree;
};
```

Each member contains/performs the following

- `int degree` - denotes the degree of the polygon
- `int * polyExpr` - dynamic array that maintains the polygon
- `polynomial::polynomial()` - default constructor that sets the degree with -1 and `polyExpr = nullptr`
- `polynomial::polynomial(const polynomial& poly)` - deep copy coonstructor (performs a deep copy of `poly.polyExpr` into `this->polyExpr`)
- `polynomial::polynomial(int * p, int degree)` - constructor that sets `this->degree = degree` and allocates `this->polyExpr = new int[degree + 1]` and assigns each element of `p` into `this->polyExpr`
- `polynomial::polynomial(int s)` - sets the `this->degree` to 0, allocates `this->polyExpr = new int[1]` and assigns the parameter into index 0
- `polynomial::~polynomial()` - destructor, deallocates `this->polyExpr`
- `polynomial polynomial::operator*(const polynomial& rhs) const` - implements the polynomial multiplication operator (when we multiply a polynomial with a polynomial)
- `polynomial polynomial::operator*(int rhs) const` - implements the polynomial multiplication operator (when we multiply a polynomial with an integer)
- `polynomial polynomial::operator+(const polynomial& rhs) const` - implements the polynomial addition operator (when we add a polynomial with a polynomial)
- `polynomial polynomial::operator+(int rhs) const` - implements the polynomial addition operator (when we add a polynomial with an integer)
- `const polynomial& polynomial::operator=(const polynomial& rhs)` - deep copy assignment operator
- `const polynomial& polynomial::operator=(int rhs)` - assignment operator that assigns an integer to a polynomial
- `polynomial polynomial::operator-() const` - unary operator, flips the sign for each coefficient of the polynomial
- `polynomial polynomial::operator-(const polynomial& rhs) const` - implements the polynomial subtraction operator (when we subtract a polynomial with a polynomial)
- `polynomial polynomial::operator-(int rhs) const` - implements the polynomial addition operator (when we add a polynomial with an integer)
- `std::ostream& operator<<(std::ostream& out, const polynomial& rhs)` - output operator, outputs the polynomial in its natural form
- `polynomial operator+(int lhs, const polynomial& rhs)` - friend function that implements an addition operator when we have an integer + polynomial
- `polynomial operator*(int lhs, const polynomial& rhs)` - friend function that implements a multiplication operator when we have an integer * polynomial
- `polynomial operator-(int lhs, const polynomial& rhs)` - friend function that implements a subtraction operator when we have an integer - polynomial

Specifications

- Make sure you program is memory leak free
- Properly document your code
- Each operator will be tested in a span of several mains, make sure you implement each operator correctly

Sample Run

```
% ./m01

result = -10x^7 + 10x^5 - 29x^4 + 12x^3 + 4x^2 - 22x + 30

% ./m02

result = 19x^5 + 3x^4 + x^3 + 4x^2 - 11


% ./m03

-result = 10x^7 - 20x^6 - 10x^5 + 24x^4 - 20x^3 + 20x^2 + 20x - 24

% ./m04

result = -5x^4 - 2x^3 + 4x^2 + 1

% ./m05

20x^4 + 8x - 24
-2x^3 + 4x^2 + 2x + 2

% ./m06

result = 15x^4 + 6x - 18
result = -2x^3 + 4x^2 + 2x - 3
result = 2x^3 - 4x^2 - 2x + 7

% ./FinalBoss

p1 = 5x^4 + x^2 + 5x + 4
p2 = 3x + 2
p3 = x^4 - 2x^3 - 3x^2 + 4x - 8
p4 = 5
p5 = 6

p1 * p2 = 15x^5 + 10x^4 + 3x^3 + 17x^2 + 22x + 8
p1 * p3 = 5x^8 - 10x^7 - 14x^6 + 23x^5 - 49x^4 - 19x^3 - 24x - 32
p1 + p2 = 5x^4 + x^2 + 8x + 6
p1 * p5 = 30x^4 + 6x^2 + 30x + 24
p5 + p3 = x^4 - 2x^3 - 3x^2 + 4x - 2
p1 * p3 + p2 = 5x^8 - 10x^7 - 14x^6 + 23x^5 - 49x^4 - 19x^3 - 21x - 30
p2 - p1 = -5x^4 - x^2 - 2x - 2
p3 - 4 = x^4 - 2x^3 - 3x^2 + 4x - 12
```

```
p1 * 3 = 15x^4 + 3x^2 + 15x + 12
p3 + 2 = x^4 - 2x^3 - 3x^2 + 4x - 6
3 - p2 = -3x + 1
2 * p1 = 10x^4 + 2x^2 + 10x + 8
5 + p2 = 3x + 7
```

Submission

Submit the source files to code grade by the deadline

References

- Supplemental Video https://youtu.be/RGlLKTEjNfw
- Link to the top image can be found at https://cdn140.picsart.com/253983425012212.png