A polynomial class

This next mini-project can be either simple or very complex, as there are many calculations that can be performed on polynomials. Also, not every class will have all possible functionality. This project will be broken into three parts:

1. First, you will be asked to define the polynomial class, meaning, you will need to identify the design of your polynomial class, you will then need to define the constructors and destructors and describe their behavior, followed by the copy and move constructors and the assignment and move operators. You will then have to decide what are relevant member functions and member operators. You may determine that there are specific operations that can only be defined outside the class (such as when the right-hand operand is an object and the left-hand operand is a primitive data type).
2. Next, we will present you with what is a reasonably comprehensive class definition as you were asked to describe in Part 1. You should ask yourself whether or not functionality you've described is really necessary (did we make a mistake), and you should ask yourself whether or not functionality we've described is really necessary (again, did we make a mistake). You can then implement any or all of the constructors, the destructor, member operators, member functions and functions defined outside the class as you wish.
3. Finally, we will give you our comprehensive implementation of the class definition that we proposed in Part 2. You can compare and contrast your implementations with ours: were you more efficient, or were we more efficient, or did we both converge on the same solution?

Now, recall that a polynomial is a linear combination[1] of powers of a variable, where a *term* is a coefficient multiplied by a power of that variable. We will use x to be that variable. For example, $3.23x$, $-4.75x^9$ and 91.35 are all terms in x, with powers 1, 9 and 0, respectively. The *degree* of a polynomial is the largest power that has a non-zero coefficient, although for simplicity, we will say that the degree of the zero polynomial is also 0. A polynomial can therefore alternatively be described as a finite sum of terms. Another description for a term is a *monomial.*

Suppose that p, q and r are polynomials in x, in which case, from secondary school, you learned that you could:

1. determine the degree of the polynomial,
2. evaluate a polynomial p at a point x; that is, calculate $p(x)$,
3. add or subtract two polynomials: $p + q$ and $p - q$,
4. add or subtract a polynomial and constant c: $p + c$, $c + p$, $p - c$, $c - p$,
5. negate a polynomial $-p$,
6. multiply a polynomial by a constant c: cp and pc,
7. multiply two polynomials: pq,
8. find the quotient and remainder when a polynomial p is divided by a divisor polynomial d, so $p = qd + r$,
9. calculating the derivative, the antiderivative and the integral of a polynomial, and

[1] A linear combination of k objects $a_1, \ldots, a_k$ is any sum of scalar multiples of these objects, where the scalars can also be zero. Linear combinations of x, y and z include $x - z$, $3.2x - 4.7y + 9.1z$, $5.4y$, $73.2x - 47.2y + z$. In the last case, the implied scalar is one (1). A linear combination of terms $1, x, x^2, \ldots, x^n$ is therefore also described as a polynomial.

10. determine if two polynomials are equal or not.

You may remember specific rules about polynomial addition and multiplication:

1. If deg(p) ≠ deg(q), then deg(p ± q) = max(deg(p), deg(q)), while if deg(p) = deg(q) then deg(p ± q) ≤ max(deg(p), deg(q)).
2. If p = 0 or q = 0, then deg(pq) = 0, otherwise deg(pq) = deg(p) + deg(q).

You may recall from calculus that the derivative and anti-derivative of the polynomial

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

are

$$n a_n x^{n-1} + (n-1) a_{n-1} x^{n-2} + \cdots + 2a_2 x + a_1$$

and

$$\frac{1}{n+1} a_n x^{n+1} + \frac{1}{n} a_{n-1} x^n + \cdots + \frac{1}{3} a_2 x^3 + \frac{1}{2} a_1 x^2 + a_0 x ,$$

respectively, although the latter usually is associated with a constant of integration (that is, a function that is mapped to the zero polynomial by the differential operator.

You may recall from calculus that the integral of a polynomial is calculated as follows:

$$\begin{aligned}
\int_a^b p(x)\,\mathrm{d}x &= \int_a^b \left(a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0\right)\mathrm{d}x \\
&= a_n \int_a^b x^n\,\mathrm{d}x + a_{n-1}\int_a^b x^{n-1}\,\mathrm{d}x + \cdots + a_2 \int_a^b x^2\,\mathrm{d}x + a_1 \int_a^b x\,\mathrm{d}x + a_0 \int_a^b 1\,\mathrm{d}x \\
&= a_n \left.\left(\frac{1}{n+1}x^{n+1}\right)\right|_a^b + a_{n-1}\left.\left(\frac{1}{n}x^n\right)\right|_a^b + \cdots + a_2 \left.\left(\frac{1}{3}x^3\right)\right|_a^b + a_1 \left(\frac{1}{2}x^2\right)_a^b + a_0 \left. x\right|_a^b \\
&= \frac{a_n}{n+1}\left(b^{n+1} - a^{n+1}\right) + \frac{a_{n-1}}{n}\left(b^n - a^n\right) + \cdots + \frac{a_2}{3}\left(b^3 - a^3\right) + \frac{a_1}{2}\left(b^2 - a^2\right) + a_0 (b - a)
\end{aligned}$$

There are many arithmetic operators in C++, some of which may or may not apply to polynomials, but here is an interesting example: if p is the polynomial such that:

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$$

then

$$p(2) = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0 .$$

You will recall that the binary number $a_n a_{n-1} \cdots a_2 a_1 a_0 = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0$. You will also recall that a binary number left-shifted by 3 is equivalent to adding three zeros:

$$a_n a_{n-1} \cdots a_2 a_1 a_0 000 = a_n 2^{n+3} + a_{n-1} 2^{n+2} + \cdots + a_2 2^5 + a_1 2^4 + a_0 2^3 .$$

Consequently, it would be completely analogous to treat a left-shift operation `(p << k)` as multiplying the polynomial by x^k and the right-shift operation `(p >> k)` as dividing the polynomial by x^k and discarding the remainder, so

$$(p \ll k)(x) = a_n x^{n+k} + a_{n-1} x^{n+k-1} + \cdots + a_2 x^{2+k} + a_1 x^{1+k} + a_0 x^k .$$

Note that during a co-operative program placement, you as an engineering student may be asked to program code without any additional help or prior teaching, so while you may never have seen integration prior to this project, you will see many more descriptions of algorithms at the workplace that you have not been taught during your undergraduate studies. You will have to look up formulas in textbooks or on-line and implement the formulas you have found (once you've determined the formulas you have found are indeed correct).

To date, the only reasonable data structure for storing a polynomial you have learned to date is using an array. We will implement this class using an array to store the coefficients.

Now, what kind of constructors would you like?

1. Define the coefficients of a polynomial by passing an array of coefficients, where `a[0]` is the constant coefficient, `a[1]` is the coefficient of x, and so on.
2. Create the binomial $a(x - b)^n$, so if $a = 1$ and $b = 0$, this creates x^n, and if $n = 0$, this creates the constant polynomial a.
3. Create the constant polynomial c.

As polynomials are described in terms of the degree, but a polynomial of degree n has $n + 1$ coefficients, then we must either store the degree or the capacity of the array. We could store both, but as one is a trivial computation of the other, this is hardly necessary and a waste of memory and a potential source of bugs (such as not always updating both member variables).

Another question you may ask yourself is will the array always have the capacity equal to the degree plus one, or will we allow a capacity that is larger than what is needed for the polynomial. In the latter case, we may not need to grow the polynomial if the degree of the polynomial is increased.