Lab 6: Polynomials

[deadline**: 18th April 2024]**

Prologue

Polynomial function is function of a single independent variable, in which that variable can appear more than once, raised to any integer power. A polynomial of degree n is a function of the form

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

where the a's are real numbers (sometimes called the coefficients of the polynomial). The degree of a polynomial is the highest power of x in its expression.

Task

Define the classes `polynomial` to store polynomial with a certain degree. Design this class so that the polynomial degree and its coefficients are non-public (coefficients keep in the table created on the heap). You must define methods that allow you to read and set these fields: define the member function to read the polynomial degree, and define the indexing operator for reading and writing coefficients. If the program attempted to set the coefficient with the highest power to 0, an exception should be raised (except when the polynomial degree is equal to 0).

```
class polynomial {
private:
    int n; // the degree of the polynomial
    double *a; // the coefficients of the polynomial
    // ...
};
```

Coefficients of the polynomial let be written in the table according to indexes, so the first coefficient a_0 is written in the cell `a[0]`.

Define a non-argument constructor and constructor with a list of coefficients in the class `polynomial`. Implement copying and moving in the class.

```
class polynomial {
public:
    polynomial(int deg=0, double coef=1.0); // monomial
    polynomial(int deg, const double coef[]);
    polynomial(initalizer_list<double> coef);
    polynomial(const polynomial &poly); // copy constructor
    polynomial(polynomial &&poly); // move constructor
    polynomial& operator = (const polynomial &poly); // copy assignment
    polynomial& operator = (polynomial &&poly); // move assignment
    ~polynomial(); // destructor
    // ...
};
```

The destructor should release the memory allocated to remember the polynomial coefficients.

Define addition, subtraction, and multiplication operators for polynomials (multiplication by constant and by another polynomial), and the function to calculating the value of polynomial at a given point using the *Horner scheme*.

```
class polynomial {
public:
    friend polynomial operator+ (const polynomial &p, const polynomial &q);
    friend polynomial operator - (const polynomial &p, const polynomial &q);
    friend polynomial operator * (const polynomial &p, const polynomial &q);
    friend polynomial operator * (double c);
    polynomial& operator += (const polynomial &q);
    polynomial& operator -= (const polynomial &q);
    polynomial& operator *= (const polynomial &q);
    polynomial& operator *= (double c);
    double operator () (double x) const; // Horner scheme
    double operator [] (int i) const; // coefficient a_i
    // ...
};
```

The polynomial class should make it easy to read the coefficients standing at monomials using the indexing operator. Remember about stream operators for reading and writing for polynomial.

Finally, write a program that reliably tests all operations on polynomials. All objects in your program should be created on the stack.

Whenever we encounter any errors, ambiguities or contradictions in the program, this should be signaled by an exception.