Problem 1

file: Polynom label: polynom

Create a class **Polynom**, object of which represent polynomials of arbitrary degrees. Coefficients of a polynomial can be represented by a **vector<double>**; its degree can be held in a separate field, although it is not necessary. Define public constructors:

- default, creating an object representing the polynomial $P(x) = 0$ of degree 0;
- taking an array of **double**s representing coefficients of the polynomial, and its size;
- a similar constructor, but taking an initializer list of **double**s;
- copy and move constructors.

Overload the `operator<<`, so objects of the class can be conveniently printed to output streams.

The class (by means its methods and friend functions) should ensure correct behavior of

- one- and two-argument operator +;
- one- and two-argument operator -;
- multiplication operator *;
- division operator /;
- remainder operator %;
- method **at(double)** returning the value of the polynomial for a given argument;
- static member
  `polydiv_t polydiv(const Polynom&, const Polynom&)`
  returning an object of type **polydiv_t**, which is a structure with two fields of type **Polynom**: quot (quotient) and rem (remainder) — see below.

Division by zero polynomial should provoke an exception of type **domain_error**. Note: Division of polynomials yields a quotient ($Q(x) = P(x)/D(x)$) and a remainder ($R(x) = P(x)\%D(x)$)) defined as usually: $P(x) = Q(x)D(x)+R(x)$, where the degree of $R(x)$ is smaller than that of $D(x)$.

Try to use facilities provided by the standard library. Do not forget about **std::move**'ing, where appropriate. For example, the program below

download *PolynomE.cpp*

```
#include <algorithm>      // copy, transform, min/max...
#include <cassert>        // may be useful for checks...
#include <cstdlib>        // abs (if needed)
#include <functional>     // plus, negate (if needed)
#include <iomanip>        // may be useful in operator<<
#include <iostream>
#include <initializer_list>
#include <numeric>        // accumulate (if needed)
```

```
#include <stdexcept>     // domain error
#include <utility>       // move
#include <vector>

struct polydiv_t;  // forward declaration

class Polynom {
    size_t deg;
    std::vector<double> coeffs;

 public:
    Polynom();
    Polynom(const double a[], size_t sz);
    Polynom(const std::initializer_list<double>& li);
    Polynom(const Polynom& p);
    Polynom(Polynom&& p) noexcept;
    Polynom& operator=(const Polynom& p) &;
    Polynom& operator=(Polynom&& p) & noexcept;
    ~Polynom();

    static polydiv_t polydiv(const Polynom& lhs,
                             const Polynom& rhs);

    Polynom operator-() const;
    Polynom operator+() const;
    Polynom operator/(const Polynom& p) const;
    Polynom operator%(const Polynom& p) const;

    double at(double x) const;

    friend Polynom operator+(const Polynom& lhs,
                             const Polynom& rhs);
    friend Polynom operator-(const Polynom& lhs,
                             const Polynom& rhs);
    friend Polynom operator*(const Polynom& lhs,
                             const Polynom& rhs);
    friend std::ostream& operator<<(std::ostream& s,
                                    const Polynom& p);
};

struct polydiv_t {
    Polynom quot;
    Polynom  rem;
};
```

```
// ...

int main() {
    using std::cout;

    double arr[] = {1, 2, 3, 4};
    Polynom p(arr, std::size(arr)), q{1, 2, 3}, r;
    cout << "p         = " << p << "\n"
         << "q         = " << q << "\n";
    Polynom ppq = p + q, ppqmp = ppq - p;
    cout << "p+q       = " << ppq   << "\n";
    cout << "p+q-p     = " << ppqmp << "\n";
    Polynom s{1, 0, 1}, prod = q * s;
    cout << "q*{1,0,1} = " << prod << "\n";
    Polynom t = (prod * r);
    cout << "Is it 0?    " << prod * r << "\n";;

    Polynom P{1, -1, 1, 1}, D{-1, 0, 2};
    polydiv_t res = Polynom::polydiv(P, D);
    cout << "P   = " << P << "\n"
         << "D   = " << D << "\n";
    cout << P << " = \n   " << D << "*" << res.quot
         << "\n + " << res.rem << "\n";
    Polynom Q{P/D};
    Polynom R{P%D};
    cout << "P/D = " << Q << "\n";
    cout << "P%D = " << R << "\n";
    cout << "Q*D+R-P = " << Q*D + R - P << "\n";

    double x{2};
    Polynom V{7};
    cout << "P(" << x << ") =  " << P.at(x) << "\n";
    cout << "V(" << x << ") =  " << V.at(x) << "\n";
}
```

should print

```
p         = [ +4x^3+3x^2+2x^1+1 ]
q         = [ +3x^2+2x^1+1 ]
p+q       = [ +4x^3+6x^2+4x^1+2 ]
p+q-p     = [ +3x^2+2x^1+1 ]
q*{1,0,1} = [ +3x^4+2x^3+4x^2+2x^1+1 ]
Is it 0?    [ 0 ]
P   = [ +1x^3+1x^2-1x^1+1 ]
D   = [ +2x^2-1 ]
[ +1x^3+1x^2-1x^1+1 ] =
```

```
   [ +2x^2-1 ]*[ +0.5x^1+0.5 ]
 + [ -0.5x^1+1.5 ]
P/D = [ +0.5x^1+0.5 ]
P%D = [ -0.5x^1+1.5 ]
Q*D+R-P = [ 0 ]
P(2) =  11
V(2) =  7
```
