Cél:

- Statikus tagok
  - `Point` osztály (4. gyakorlat)
- Move constructor & Move assignment
  - `Matrix` (5. gyakorlat)
  - `Polynomial`
- Operátorok túlterhelése:
  - `Polynomial`

1. feladat

Egészítse ki a 4. gyakorlatot (**Projekt - lab04**)

Egészítse ki a `Point` osztályt úgy, hogy az osztály képes legyen a saját példányszámának nyilvántartására.

Minden konstruktorban növelje a counter értéket, illetve a destruktorban csökkentse ezt.

```
class Point{
private:
   int x, y;
   static int counter;
public:
   Point( int x=0, int y=0);
   int getX() const;
   int getY() const;
   double distanceTo(const Point& point) const;
   Point(const Point&);
   ~Point();
   static int getCounter();
};
```

A statikus adattagot nem lehet deklarációban inicializálni, ezt végezze a Point.cpp állományban a következőképpen:

```
int Point::counter{0};
```

Egészítse ki a main metódust a következő kódrészlettel:

```
PointSet pset1(10);
cout<<"#points: " <<Point::getCounter()<<endl;
PointSet pset2(20);
cout<<"#points: " <<Point::getCounter()<<endl;
```

2. feladat

Implementálja az 5. gyakorlaton kért move konstruktort és move assignment operátort (**Projekt - lab05**). Tesztelje ezeket a metódusokat!

3. feladat - Parciális 2017

Adott egy **Polynomial.h** fejállomány, amely egy egész együtthatós polinom típust deklarál.

```
#ifndef POLYNOMIAL_H
#define POLYNOMIAL_H

#include <iostream>
using namespace std;

class Polynomial {
   // Polinom együtthatói
   double *coefficients;
   // Polinom együtthatóinak a száma
   int capacity;
public:
   Polynomial(int degree, const double coefficients[]);
   Polynomial(const Polynomial &that);
   Polynomial(Polynomial &&that);
   ~Polynomial();
   // Polinom fokszáma
   int degree() const;
   // Polinom értéke a megadott pontban
   double evaluate(double x) const;
   // Polinom deriváltja
   Polynomial derivative() const;
   double operator[](int index) const;
   friend Polynomial operator -(const Polynomial &a);
   friend Polynomial operator +(const Polynomial &a, const Polynomial &b);
   friend Polynomial operator -(const Polynomial &a, const Polynomial &b);
   friend Polynomial operator *(const Polynomial &a, const Polynomial &b);
   friend ostream & operator <<(ostream& out, const Polynomial& what);
   /* copy assignment - mély másolat letiltása értékadásra nézve */
   Polynomial& operator=(const Polynomial&) = delete;
   /* move assignment - sekély másolat letiltása értékadásra nézve */
   Polynomial& operator=(Polynomial&&) = delete;
};
#endif
```

FEJLETT PROGRAMOZÁSI TECHNIKÁK (C++)
6. GYAKORLAT

Követelmény

- Implementáljuk a deklarált függvényeket / operátorokat egy **polynomial.cpp** állományban.
- Minden egyes függvényre készítsünk olyan kódrészletet (**main.cpp**), amelyben ellenőrizhető az adott függvény helyessége.

Magyarázat

- A degree függvény visszatéríti a polinom fokszámát.
- Az evaluate függvény kiértékeli a polinomot a megadott pontban Horner-módszerrel.
- A derivative függvény visszatéríti a polinom derivált polinomját.
- Az index operatort elegendő csak olvasásra megírni.

Pontozás:

- konstruktor: 0.5 pont
- copy konstruktor: 0.5 pont
- move konstruktor: 0.5 pont
- destruktor: 0.5 pont
- operator<<: 1 pont
- degree(): 0.5 pont
- evaluate(): 0.5 pont
- derivative(): 0.5 pont
- operator[]: 1 pont
- operator -, +, -, *: 2 pont
- main: 2.5 pont