3. Kis zárthelyi, megoldások és pontozás

a kijavított kis zárthelyik megtekinthetők az április 10. gyakorlat után

---

1. Igaz állítások, egészítse ki stb. a válasz egyértelmű…

---

2. Írjon 2D-s Pont osztályt! (7 pont) Az osztály kívülről hozzáférhetően tárolja a pont x és y értékét (double x, y)! A ponthoz lehessen hozzáadni, belőle kivonni más kétdimenziós pontot! Két pontról legyen eldönthető, hogy megegyeznek-e (megegyeznek, ha koordinátáik egyenlők)! Ehhez a következő operátorokat írjuk felül: operator+, operator- és operator==! A tagfüggvényeket az osztályon belül írjuk meg.
1 pont: helyes osztály 2 pont: helyes deklarációk + op fejlécek 1 pont: helyes konstruktor 1 pont: helyes op+ 1 pont: helyes op== 1 pont: helyes op-

Egy lehetséges megoldás:

```
class Point
{
public:
	double x;
	double y;

	Point(double x, double y) : x(x), y(y){};

	Point operator+(const Point &p) const
	{
		return Point(x + p.x, y + p.y);
	}

	Point operator-(const Point &p) const
	{
		return Point(x - p.x, y - p.y);
	}

	bool operator==(const Point &p) const
	{
		return x == p.x && y == p.y;
	}

};
```

Megjegyzések

- A kétoperandusú aritmetikai operátorok mindig új példánnyal kell, hogy visszatérjenek, az eredeti példány adattagjai nem módosulhatnak. Ha az eredeti példányt módosítjuk, az += stb. jellegű művelet lesz.

```
Point &operator+=(const Point &p)
{
	x += p.x;
	y += p.y;
	return *this;
}
```

- Mikor használunk referenciát? Lefordul referencia nélkül is és általában működőképes.
  - paramétert, ha csak olvassuk, érdemes konstans referenciaként átadni. Egyrészt nem foglalunk helyet a veremben, másrészt megtakarítunk egy másolást. Mivel nem módosítunk rajta, az információ jó helyen van az eredeti helyén, cím szerint adjuk át, teljesen felesleges egy lokális másolatot készíteni.
  - A visszatérési érték akkor legyen referencia, ha a példány módosult, azaz a += jellegű műveletek. Ekkor önmagát kell, hogy visszaadja a példány, *this
  - lokálisan létrehozott változó referenciáját ne adjuk vissza, öngyilkosság...
  - Írjuk meg a copy constructor-t loggolósra, akkor látni fogjuk, hol történik másolás...

```
	Point(const Point &p)
	{
		*this = p;
		std::cerr << "Copy ctr" << std::endl;
	}
```