rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 2. beszámoló dolgozat. 2017.03.03. Kurz/**Terem**: Gy3/	Elért pontszám:
Név: **Neptun:**	

1. Feladat: (10p)

a) Valósítsa meg C++ nyelven a *Macska* osztályt! Az osztálynak legyen egy egész attribútuma és egy valós állandója! Ez utóbbi a macska fekete szőrszálainak arányát adja meg és nem változtatható meg (konstans). Macska két módon jöhet létre: konstruktorhívással vagy két macska találkozásával. Legyen az osztálynak:

- Olyan **konstruktora**, amivel létrehozható egy tetszőleges korú és szőrszálállandójú macska. Ez a konstruktor legyen 1 paraméterrel is hívható, ekkor állítsa 0-ra a kort, a szőrszálállandóját pedig a paraméterként adott értékre!
- Olyan **tagfüggvénye** (getKor), amivel le lehet kérdezni a macska korát!
- Olyan **tagfüggvénye** (getSzal), amivel le lehet kérdezni a szőrszálállandót!
- Olyan **tagfüggvénye** (pre inkremens operátor), amivel a kor megnövelhető.

Amennyiben egy macska **elpusztul**, utolsó leheletéből írja ki, az szabványos kimenetre, hogy „nyau”.
Két macska találkozását az **összeadás** (+) operátorral modellezze! Ekkor egy 0 éves új macska keletkezik, aminek a szőrszálállandója a két macska szőrszálállandójának átlaga.
Készítsen olyan **inserter** (<<) operátort is, amivel egy **std::ostream** típusú objektumra a macska kora és szőrszálállandója kiírható. Az osztály adattagjai ne legyenek kívülről közvetlenül elérhetők! Ügyeljen arra, hogy a megfelelő függvények konstans macskákon is meghívhatók legyenek! Az osztály használatát az alábbi kódrészlet szemlélteti:

```
{
    Macska m1(0.7, 1);
    Macska m2(0.1, 2);
    Macska m3 = m1 + m2;
    std::cout << m3 << std::endl;      // Macskaszor: 0.4 Kora: 0.........
    ++m3;
    std::cout << m3 << std::endl;      // Macskaszor: 0.4 Kora: 1.........
}                                      // nyaunyaunyau.....................
```

b) Miután elkészült az osztállyal, írja a pontozott vonalra, hogy mit ír ki az adott helyen a program a szabványos kimenetre

```
class Macska {
    const double szal;
    int kor;
public:
    Macska(double szal, int kor = 0) :szal(szal), kor(kor) {}
    double getSzal () const { return szal; }
    int getKor() const { return kor; }
    Macska operator+(const Macska& macsek) const {
        return Macska((szal+macsek.szal)/2);
    }
    Macska& operator++() { ++kor; return *this; }
    ~Macska() { std::cout << "nyau"; }
};

std::ostream& operator<<(std::ostream& os, const Macska& m) {
    return os << "Macskaszor: " << m.getSzal() << " Kora: " << m.getKor();
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 1. beszámoló dolgozat. 2017.03.03. Kurz/**Terem**: Gy3	Elért pontszám:
Név: **Neptun:**	

1. Feladat: (10p)

a) Valósítsa meg C++ nyelven a Nyúl osztályt! Az osztálynak legyen egy egész attribútuma és egy float állandója! Ez utóbbi a nyúl bátorságát fejezi ki és nem változtatható meg (konstans). Nyúl objektum két módon jöhet létre: konstruktorhívással vagy két nyúl objektum találkozása révén. Legyen az osztálynak:

- Olyan **konstruktora**, amivel létrehozható egy tetszőleges korú, és bátorságú nyúl. Ez a konstruktor legyen 1 paraméterrel is hívható, ekkor állítsa 1-re a kort, a bátorságot pedig a paraméterként adott értékre!
- Olyan **tagfüggvénye** (getKor), amivel le lehet kérdezni a nyúl korát!
- Olyan **tagfüggvénye** (getBator), amivel le lehet kérdezni a bátorságát!
- Olyan **tagfüggvénye** (pre inkremens operátor), amivel a kor megnövelhető.

Amennyiben egy nyúl **elpusztul**, utolsó leheletéből írja ki, az szabványos kimenetre, hogy „Hopp”.
Két nyúl találkozását a **szorzás**(*) operátorral modellezze! Ekkor egy 0 éves új nyúl keletkezik, aminek a bátorsága azonos az operátor jobb oldalán álló nyúl bátorságával.
Készítsen olyan **inserter** (<<) operátort is, amivel egy **std::ostream** típusú objektumra kiírható egy nyúl kora és bátorsága. Az osztály adattagjai ne legyenek kívülről közvetlenül elérhetők! Ügyeljen arra, hogy a megfelelő függvények konstans objektumokon is meghívhatók legyenek! Az osztály használatát az alábbi kódrészlet szemlélteti:

```
{
    Nyul nyuszi1(1, 2);
    Nyul nyuszi2(0.4, 4);
    Nyul gyerek = nyuszi1 * nyuszi2;
    std::cout << gyerek << std::endl;  // Bator: 0.4 Kora: 0..............
    ++gyerek;
    std::cout << gyerek << std::endl;  // Bator: 0.4 Kora: 1..............
}                                      // HoppHoppHopp.....................
```

b) Miután elkészült az osztállyal, írja a pontozott vonalra, hogy mit ír ki az adott helyen a program a szabványos kimenetre

```
class Nyul {
    const float bator;
    int kor;
public:
    Nyul(double bator, int kor = 1) :bator(bator), kor(kor) {}
    double getBator() const { return bator; }
    int getKor() const { return kor; }
    Nyul operator*(const Nyul& nyuszkany) const {
        return Nyul(nyuszkany.bator, 0);
    }
    void operator++() { ++kor; }
    ~Nyul() { std::cout << "Hopp"; }
};

std::ostream& operator<<(std::ostream& os, const Nyul& ny) {
    return os << "Bator: " << ny.getBator() << " Kora: " << ny.getKor();
}
```