rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.24. Kurz/**Terem**: Gy3/	Elért pontszám:
Név: **Neptun:**	

1. Jelölje (pl. karikázza be), hogy az állítás igaz (I), vagy hamis (H) a C++ nyelvre! Minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív. (4p)

Az inicializálás és az értékadás ugyanaz.	I	**H**
Statikus tagfüggvénynek nincs this pointere.	I	H
A destruktor mindig meghívja a tartalmazott objektumok destuktorait.	I	H
A konstruktor előbb hajtja vére a programozott törzset, és csak ezután hívja a tartalmazott objektumok konstruktorait.	I	H

2. A *Szamlalo* osztály egy egyszerű számlálót modellez. A *clock* tagfüggvény meghívásakor a számláló eggyel növekszik. Az osztály felhasználásával, annak módosítása nélkül hozzon létre egy *Jelzo* osztályt, ami a *Szamlalo* osztályhoz hasonlóan működik, de a *clock* tagfüggvénye a konstruktorban megadott számlálóállásnál „jelez”, azaz amikor a számláló eléri a konstruktorban megadott értéket, a szabványos kimenetre kiírja, hogy „BRRR”! (6p)

```
class Szamlalo {
    int cnt;
public:
    Szamlalo() :cnt(0) { }
    int getCnt() const { return cnt; }
    void setCnt(int n) { cnt = n; }
    void clock()       { cnt++; }
};
```

A *Jelzo* osztályt úgy valósítsa meg, hogy az legyen kompatibilis a *Szamlalo* osztállyal! Valósítson meg minden olyan osztályt és függvényt, ami az alábbi kódrészlet működéséhez kell! A komment segít megérteni az elvárt működést és az elvárt kimenetet.

```
    std::cout << Szamlalo() << std::endl;// Kiír: 0
    Jelzo j1(10);                        // Létrejött egy jelző, ami 10-nél jelez
    std::cout << j1 << std::endl;        // A számláló lekérdezése. Kiír: 0
    j1.clock();                          // Növeljük a számlálót
    std::cout << j1 << std::endl;        // Számláló növekedett. Kiír: 1
    j1.setCnt(9);                        // Kilenc értékűre állítjuk a számlálót
    j1.clock();                          // Kiír: BRRR
    j1.clock();                          // Növeljük a számlálót
    std::cout << j1 << std::endl;        // Kiír: 11
```

```
std::ostream& operator<<(std::ostream& os, const Szamlalo& sz) {
    os << sz.getCnt();
    return os;
}

class Jelzo : public Szamlalo {
    int alarm;
public:
    Jelzo(int alarm) :alarm(alarm) {}
    void clock();
};

void Jelzo::clock() {
    Szamlalo::clock();
    if (getCnt() == alarm) std::cout << "BRRR" << std::endl;
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.24. Kurz/**Terem**: Gy3/	Elért pontszám:
Név: **Neptun:**	

1. Jelölje (pl. karikázza be), hogy az állítás igaz (I), vagy hamis (H) a C++ nyelvre! Minden bejelölt válasz 1 pont, ha helyes, -1 pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív. (4p)

A statikus adattag a memóriában csak egyetlen egyszer szerepel, és az osztályhoz tartozik.	**I**	H
Az operator+= automatikusan keletkezik, ha van operator+	I	**H**
A throw utasítás értéket dob, ezért másoló konstruktort hívhat.	**I**	H
Virtuális függvény nem lehet konstans	I	**H**

2. A Counter osztály egy egyszerű számlálót modellez. Az *incr* tagfüggvény meghívásakor számláló eggyel növekszik. Az osztály felhasználásával, annak módosítása nélkül hozzon létre egy *Signal* osztályt, ami *Counter* osztályhoz hasonlóan működik, de az *incr* tagfüggvénye a konstruktorában megadott számlálóállásnál „jelez”, azaz amikor a számláló eléri a konstruktorban megadott értéket, a szabványos kimenetre kiírja, hogy „HAHO”! (6p)

```
class Counter {
    int cnt;
public:
    Counter() :cnt(0)  { }
    int getCnt() const { return cnt; }
    void setCnt(int n) { cnt = n; }
    void incr()        { cnt++; }
};
```

A *Signal* osztályt úgy valósítsa meg, hogy az legyen kompatibilis a *Counter* osztállyal! Valósítson meg minden olyan osztályt és függvényt, ami az alábbi kódrészlet működéséhez kell! A komment segít megérteni az elvárt működést és az elvárt kimenetet.

```
    std::cout << Counter() << std::endl; // Kiír: 0
    Signal s1(8);                        // Létrejött egy jelző, ami 8-nál jelez
    std::cout << s1 << std::endl;        // A számláló lekérdezése. Kiír: 0
    s1.incr();                           // Növeljük a számlálót
    std::cout << s1 << std::endl;        // Számláló növekedett. Kiír: 1
    s1.setCnt(7);                        // 7-re állítjuk a számlálót
    s1.incr();                           // Kiír: HAHO
    s1.incr();                           // Növeljük a számlálót
    std::cout << s1 << std::endl;        // Kiír: 9
```

```
std::ostream& operator<<(std::ostream& os, const Counter& sz) {
    os << sz.getCnt();
    return os;
}

class Signal : public Counter {
    int alarm;
public:
    Signal(int alarm) :alarm(alarm) {}
    void incr();
};

void Signal::incr() {
    Counter::incr();
    if (getCnt() == alarm) std::cout << "HAHO" << std::endl;
}
```