rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. beszámoló dolgozat. 2017.02.24. Kurz/**Terem**: Gy3/	Elért pontszám:
Név: **Neptun:**	

Az első feladatnál 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.

1. Feladat (4p)

Jelölje, hogy mely kijelentés(ek) igaz(ak) a C++ nyelvre

- ☐ **`inline`** függvényben nem lehet **`for`** utasítás.
- ☐ A referencia egy alternatív név.
- ☐ **`realloc`** helyett **`renew`** utasítást kell használni.
- ☐ A konstruktor mindig **`int`** típusú, amit nem kell kiírni.
- ☐ Minden C++ program a **`#include <iostream>`** direktívával kell, hogy kezdődjön.
- ☐ Konstans tagfüggvény nem változtathatja meg az objektum állapotát.

Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre!

```
struct S {
   int *a;
   S()  { this->a = 0; }
   ~S() {}
};
```

- ☐ **`S`** egy osztály.
- ☐ **`S`** egy objektum.
- ☐ **`S`**-nek nincs konstruktora.
- ☐ Memóriaszivárgás lép fel.
- ☐ A deklaráció hibás, mert a **this** csak akkor használható, ha van a tagfüggvénynek paramétere.
- ☐ Minden adat és tagfüggvény privát.
- ☐ Minden adat és tagfüggvény publikus.

2. **Tervezzen** egy osztályt (Tort) törtek tárolására! Az osztály a törtet két egész számként (számláló, nevező) tárolja! Az osztálynak (6p)

- legyen paraméter nélkül hívható konstruktora, ami a számlálót 0-ra, a nevezőt 1-re állítja;
- legyen 1 paraméterrel hívható konstruktora, ami a számlálót a megadott értékre, a nevezőt 1-re állítja;
- legyen 2 paraméterrel hívható konstruktora, ami mind a számlálót, mind a nevezőt beállítja;
- legyen olyan lekérdező függvénye, amivel a tört értéke valós számmá konvertálva lekérdezhető;
- legyen olyan operátora (/), amivel egy ilyen tört egy egész számmal eloszthatóó. Az eredmény egy Tort!

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. Ügyeljen arra, hogy konstans objektumokra is jól működjön az osztály! Nullával való osztás (/) esetén az operator kivételként dobjon egy olyan törtet (Tort), amiben a számláló és a nevező is nulla!

Működjön helyesen az alábbi kódrészlet:

```
Tort t0, t1(1), t05(1, 2);                    // 0/1, 1/1, 1/2
std::cout << t2.get() << std::endl;           // Kiír: 0.5
std::cout << (t05 / 2).get() << std::endl;    // Kiír: 0.25
```

Egy lehetséges megoldás:

```
class Tort {
    int szamlalo;
    int nevezo;
public:
    class Hiba {};
    Tort(int szamlalo=0, int nevezo=1) :szamlalo(szamlalo), nevezo(nevezo) {}
    double get() const { return (double)szamlalo / nevezo; }
    Tort operator/(int) const;
};
Tort Tort::operator/(int n) const {
    if ( n == 0 ) throw Tort(0,0);
    return Tort(szamlalo, n*nevezo);
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. beszámoló dolgozat. 2017.02.24. Kurz/**Terem**: Gy3/	Elért pontszám:
Név: **Neptun:**	

Az első feladatnál 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.

3. Feladat (4p)

Jelölje, hogy mely kijelentés(ek) igaz(ak) a C++ nyelvre

- ☐ Egy osztálynak több destruktora is lehet.
- ☐ **inline** függvényben nem lehet ciklus.
- ☐ Minden osztálynak van konstruktora
- ☐ A **new** által allokált hely felszabadítható **delete[]**-tel is.
- ☐ A **private** kulcsszó osztályon belül csak egyszer szerepelhet.
- ☐ A függvények felüldefiniálhatók (fv. overload)

Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre!

```
class Z {
   int a;
public: Z(int i) { a = i; }
   ~Z() {}
};
```

- ☐ Z egy objektum.
- ☐ Z-nek nincs konstruktora.
- ☐ Z egy változó
- ☐ Z-ből nem hozható létre tömb.
- ☐ Minden adat és tagfüggvény privát.
- ☐ Minden tagfüggvény publikus.

1. **Tervezzen** egy olyan osztályt (Ar), ami árak eltérő pénznemben történő tárolására alkalmas! Az ár mellett egy árfolyamot is tárolni kell az osztályban, amit egyes műveletekben figyelembe kell venni. Mind az árat, mind az árfolyamot valós számként tárolja! Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami az árat 0-ra, az árfolyamot 1-re állítja;
   - legyen 1 paraméterrel hívható konstruktora, az árat a megadott értékre, az árfolyamot pedig 1-re álltja;
   - legyen 2 paraméterrel hívható konstruktora, ami mind az árat, mind az árfolyamot beállítja;
   - legyen olyan lekérdező függvénye, ami az ár árfolyammal megszorzott értékét adja;
   - legyen olyan lekérdező függvénye, amivel az árfolyam kiolvasható;
   - legyen olyan operátora (-), amivel két ár különbségét lehet képezni. Az eredmény egy olyan ár objektum legyen, amiben a szorzó 1, az ár pedig a két ár árfolyamhelyes különbsége!

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. Ügyeljen arra, hogy konstans objektumokra is jól működjön az osztály! A kivonást elvégző operátor kivételként dobja el a kivonás eredményét (double), ha az negatív!

   Működjön helyesen az alábbi kódrészlet:

```
Ar a0, a1Ft(1), a1Eur (1, 310);                    // 0Ft, 1Ft, 1EUR
std::cout << a1Eur.getAr() << std::endl;           // Kiír: 310
std::cout << (a1Eur - a1Ft).getAr() << std::endl;  // Kiír: 309
```

Egy lehetséges megoldás:

```
class Ar {
    double ar;
    double arfolyam;
public:
    Ar(double ar = 0, double arfolyam = 1) :ar(ar), arfolyam(arfolyam) {}
    double getAr() const { return ar * arfolyam; }
    double getArfolyam() const { return arfolyam; }
    Ar operator-(const Ar&) const;
};
Ar Ar::operator-(const Ar& rhs) const {
    double tmp = getAr() - rhs.getAr();
    if ( tmp < 0 ) throw tmp;
    return Ar(tmp, 1);
}
```