L01: csütörtök 10-12 rendelkezésre álló idő:15 **perc**

Programozás alapja 2.	1. beszámoló dolgozat. Kurz/**Terem**: L01/	2018.03.01.	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.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

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

- Minden programot `using namespace std;` direktívával kell kezdeni.
- Referencia típusú változó nem adható át paraméterként.
- A **new** képes kivételt generálni.
- A névterek egymásba ágyazhatók.
- Függvényprototípus használata kötelező, ha a függvény a használat előtt nincs definiálva.
- A **throw** utasítás végtelen ciklusból való kilépésre való.
- Kivételt csak abban a fájlban lehet elfogni, ahol a **throw** utasítás van.
- Minden programot **#include <iostream>** direktívával kell kezdeni.

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

```
{ int i; while (std::cin >> i); }
```

- Addig olvas be egész számokat, amíg az inputon egész formátumnak megfelelő adatok érkeznek
- A kódrészlet hibás, mert az **i** változó nem kap értéket.
- Végtelen ciklus.

2. **Tervezzen** egy olyan osztályt (**Ido**), ami idő (óra, perc) tárolására alkalmas! Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami 12 óra 0 percet állít be;
   - legyen olyan konstruktora is, amivel az óra és a perc is beállítható; amennyiben csak az órát adják meg paraméterként, úgy adott óra 0 percet állítson be a függvény;
   - legyen olyan operátora (+), amivel egy időponthoz egész (int) órát lehet adni jobbról! Az eredmény az összeadás műveletnél megszokott módon új objektumban keletkezzen! Amennyiben az összeadás során keletkező objektumban az óra értéke meghaladná a 24 órát, úgy dobjon **const char*** kivételt!

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
   Működjön helyesen az alábbi kódrészlet:

```
Ido t0, t10(10), t23(23, 05); // 12:00, 10:00, 23:05 időpontokat állít
t0 = t10 + 1;                 // t0-ba 11:00 kerül, t10 változatlan marad
t23 + 2;                      // kivétel keletkezik
```

```
class Ido {
    int ora;
    int perc;
public:
    Ido(int o = 12, int p = 0) :ora(o), perc(p) {}
    Ido operator+(int rhs) const;
};
Ido Ido::operator+(int rhs) const {
    if (ora + rhs > 24) throw "Ora > 24";
    return Ido(ora + rhs, perc);
}
```

L02: csütörtök 15-17 rendelkezésre álló idő:15 **perc**

Programozás alapja 2.	1. beszámoló dolgozat. Kurz/**Terem**: L02/	2018.03.01.	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.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

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

- A struktúra egy osztály.
- A **new** sosem generál kivételt.
- Az értékadó operátor nem terhelhető túl.
- A **delete[]** egy operátor.
- Minden függvény megkapja a **this** pointert egy rejtett paraméterként
- A névterek nem ágyazhatók egymásba.
- Referencia típusú változó átadható paraméterként.
- Egy változót többször is lehet definiálni, de deklarálni csak egyszer lehet.

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

```
void swap(int& a, int& b) {
    int& c = a;
    a = b;
    b = c; }
```

- A függvény felcseréli a referencia paraméterként kapott két változó adatát.
- A függvény az első paraméterként kapott változóba másolja a második paraméter adatát.
- A függvény hibás, mert lokális változó referenciáját képzi.

2. **Tervezzen** egy olyan osztályt (Pont), ami olyan síkbeli pont koordinátáinak (X,Y) tárolására való, ami az origó középpontú egységkörön belülre esik. Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami mindkét koordinátát nullára állítja;
   - legyen olyan konstruktora is, amivel mind az X, mind az Y koordináta beállítható; amennyiben csak egy paraméterrel hívják, csak az X értéket állítsa, az Y koordináta legyen 0;
   - legyen olyan operátora (*), amivel meg lehet szorozni egy pontot egy valós számmal jobbról! A szorzás során mind az X, min az Y koordináta szorzódik. Az eredmény a szorzás műveletnél megszokott módon új objektumban keletkezzen! Amennyiben a szorzás során keletkező új pont kívül esne az egységkörön, úgy dobjon **const char*** kivételt!

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
   Működjön helyesen az alábbi kódrészlet:

```
Pont p0, p10(.1), p13(0.1, 0.3);// (0,0) (0.1,0), (0.1,0.3) pontokat hoz lére
p0 = p10 * 2;                   // p0-ba (0.2,0) kerül, p10 változatlan marad
p0 = p13 * 10;                   // kivétel keletkezik
class Pont {
    double x, y
public:
    Pont(double x = 0, double y = 0) :x(x), y(y) {}
    Pont operator*(double rhs) const;
};
Pont Pont::operator*(double rhs) const {
    double _x = x*rhs;
    double _y = y*rhs;
    if ((_x *_x + _y *_y) > 1) throw "Pont > 1";
    return Pont(_x, _y);
}
```

L03: péntek 10-12 rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2018.03.02. Kurz/**Terem**: L03/	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.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

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

- `inline` függvénynek nem lehet default paramétere.
- A függvények egymásba ágyazhatók. A belső függvények a scope `(::)` operátorral érhetők el.
- **realloc** helyett **renew** utasítást kell használni.
- A ciklus feltételében is lehet változót deklarálni. Pl: `while (int i = f())`…
- Destruktornak csak konstans paramétere lehet.
- Konstans tagfüggvény nem változtathatja meg az objektum állapotát.

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

```
{ struct S { int a; }; S so; }
```

- `S` egy osztály.
- `so` destruktora sosem hívódik meg, mert nincs.
- `S` egy objektum.
- `so` adattagja inicializálatlan.
- `S` minden adattagja privát

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 olyan konstruktora is, amivel a számláló és a nevező is beállítható; amennyiben csak a számlálót adják meg, úgy a nevező legyen 1;
- legyen olyan operátora (/), amivel egy ilyen tört egy egész számmal eloszthető! Az eredmény az osztás műveletnél megszokott módon új objektumban keletkezzen! Nullával való osztás esetén dobjon **const char*** kivételt!

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni.
Működjön helyesen az alábbi kódrészlet:

```
Tort t0, t1(1), t05(1, 2);          // 0/1, 1/1, 1/2
t0 = t1 / 2;                        // t0-ba 1/2 kerül, t1 változatlan marad
t05 = t05 / 0;                      // kivétel keletkezik
```

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

L04: péntek 12-14 rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2018.03.02. Kurz/**Terem**: L04/	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.
Labor e.d. (beugró): 1. feladatból min 2 pont.

1. Feladat (4p)

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

- A `public` kulcsszó osztályon belül csak egyszer szerepelhet.
- A `new` képes kivételt generálni.
- `inline` függvényben nem lehet ciklus.
- A scope `(::)` operatorral a privát adatok is elérhetők bárki számára.
- Túlterheléssel az operátorok szintaxisa nem változtatható meg.
- A névterek nem ágyazhatók egymásba.

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

```
{ class C { int x; }; C co;}
```

- `C` egy objektum.
- `C` egy osztály.
- `C` minden adattagja privát.
- C konstruktora nem hívódik meg, mert nincs
- A kódrészletben memóriaszivárgás lép fel.

2. Tervezzen egy olyan osztályt (Vektor), ami síkbeli pontok helyvektorának tárolására használható! A helyvektort X,Y koordinátákkal tároljuk. Az osztálynak (6p)

- legyen paraméter nélkül hívható konstruktora, ami mindkét koordinátát nullára állítja;
- legyen olyan konstruktora is, amivel mind az X, mind az Y koordináta beállítható; amennyiben csak egy paraméterrel hívják, úgy az X értéket állítsa az adott értékre, Y legyen 0;
- legyen olyan operátora (/), amivel egy vektort el lehet osztani egy egész számmal! Az osztás során mind az X, min az Y koordináta osztódik. Az eredmény az osztás műveletnél megszokott módon új objektumban keletkezzen! Nullával való osztás esetén dobjon **const char*** kivételt!

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. **Működjön** helyesen az alábbi kódrészlet:

```
Vektor v0, v1(2), v15(1, 5);        // (0, 0); (2,0); (1,5)
v0 = v1 / 2;                        // v0-ba (1,0) kerül, v1 változatlan marad
v15 = v15 / 0;                      // kivétel keletkezik
```

```
class Vektor {
    double x;
    double y;
public:
    Vektor(double x = 0, double y = 0) :x(x), y(y) {}
    Vektor operator/(int rhs) const;
};
Vektor Vektor::operator/(int rhs) const {
    if (rhs == 0) throw "div 0";
    return Vektor(x/rhs, y/rhs);
}
```