AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2015.03.04. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

Az első két feladatnál a helyes válasz 1 pont, a hibás -1pont! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.

1. Jelölje, hogy mely kijelentés(ek) igaz(ak) C++ nyelven a kivételkezelésre! (1p)

   ☐ Végtelen ciklusból való kilépéshez használják.
   ☐ Csak hibás programban keletkeznek kivételek, ezért nem kell velük foglalkozni.
   ☐ `for` ciklusban nem használható.
   ☐ Csak abban a fájlban lehet elfogni, ahol a throw utasítás van.
   ☐ A kivétel típus alapján kapható el.

2. Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre! (2p)

```
{ const int* p; ... p[0] = 0; p++; int& r = p[0]; ...}
```

   ☐ **p** deklarációja hibás, mert nem kap kezdőértéket.
   ☐ A **p++;** utasítás hibás, mert **p** értéke nem változtatható meg.
   ☐ **p** konstans területre mutat, melynek tartalma nem változtatható meg.
   ☐ A **p[0]** kifejezés hibás, mert pointer nem indexelhető, ***p-t** kell írni helyette.
   ☐ Az **int& r = p[0];** utasítás hibás, mert konstansra csak konstans referencia hivatkozhat.
   ☐ Hibás, mert referenciát csak globális változóra lehet létrehozni.

3. Racionális számokat egy struktúrában úgy tárolunk, hogy egész számként tároljuk a számlálót és a nevezőt:

```
struct Rac { int szamlalo; int nevezo; };
```

   Készítsen C++ nyelven a számok kiíratásához olyan függvényt, amivel azok kiírhatók a megszokott módon egy `ostream` típusú objektumra. Működjön helyesen az alábbi kódrészlet:

```
Rac r1 = { 3, 4 }, r2 = { 8, 9 };
std::cout << "r1:" << r1 << " r2:" << r2 << std::endl; //r1:3/4 r2:8/9
```

   Úgy írja ki a két értéket, hogy a számláló és a nevező közé tegyen valamilyen nyomtatható elválasztó karaktert (pl. 3/4, vagy 3:4, stb.)! Ha elkészült, írja a fenti utasítás után a vonalra, hogy megvalósítása szerint mi jelenik meg a standard kimeneten az utasítás hatására! (3p)

4. **Készítsen** C++ nyelven a racionális számok beolvasásához olyan függvényt, amivel azok a fenti formátumban megadva beolvashatók a megszokott módon egy `istream` típusú objektumról. Helyes bemeneti sorozat esetén működjön helyesen az alábbi kódrészlet, azaz fájl végéig olvasson: (4p)

```
Rac r;
while (std::cin >> r) { std::cout << r; }
```

3. Feladat megoldása:

```
std::ostream& operator<<(std::ostream& os, const Rac& r) {
    return os << r.szamlalo << "/" << r.nevezo;
}
```

4. Feladat megoldása:

```
std::istream& operator>>(std::istream& is, Rac &r) {
    is >> r.szamlalo;
    return is.ignore(1) >> r.nevezo;
}
```

BBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2015.03.04. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

Az első két feladatnál a helyes válasz 1 pont, a hibás -1pont! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott pontokkal. A teljes dolgozat eredménye azonban nem lehet negatív.

1. Jelölje, hogy mely kijelentés(ek) igaz(ak) C++ nyelven a kivételkezelésre! (1p)

   ☐ Kivételkezelésnek azt nevezzük, amikor egy formális paramétert nem nevezünk meg, mert még nem használjuk.

   ☐ Végtelen ciklusból való kilépéshez használják.

   ☐ Csak hibás programban keletkeznek kivételek, ezért nem kell velük foglalkozni.

   ☐ Csak abban a fájlban lehet elfogni, ahol a `throw` utasítás van.

   ☐ Ha elfogy a dinamikus memória, alapesetben `bad_alloc` kivétel keletkezik.

2. Jelölje, hogy mely kijelentés(ek) igaz(ak) az alábbi C++ kódrészletre! (2p)

```
{ int* const p; ... p[0] = 0; p++; int& r = p[0]; ...}
```

   ☐ **p** deklarációja hibás, mert nem kap kezdőértéket.

   ☐ **p** konstans területre mutat, melynek tartalma nem változtatható meg.

   ☐ Az **int& r = p[0];** utasítás hibás, mert konstansra csak konstans referencia hivatkozhat.

   ☐ A **p++;** utasítás hibás, mert **p** értéke nem változtatható meg.

   ☐ A **p[0]** kifejezés hibás, mert pointer nem indexelhető, ***p-t** kell írni helyette.

   ☐ Hibás, mert referenciát csak globális változóra lehet létrehozni.

3. Síkbeli koordinátákat egy struktúrában a következőképpen tárolunk:

```
struct Koord { double x; double y; };
```

   Készítsen C++ nyelven a koordináták kiíratásához olyan függvényt, amivel azok kiírhatók a megszokott módon egy `ostream` típusú objektumra. Működjön helyesen az alábbi kódrészlet:

```
Koord k1 = { 3.1, 0 }, k2 = { 0.5, 9.2 };
std::cout << "k1:" << k1 << " k2:" << k2 << std::endl; //k1:3.1;0 k2:0.5;9.2
```

   Úgy írja ki a két értéket, hogy a két koordináta közé tegyen vesszőt, vagy pontosvesszőt! Ha elkészült, írja a fenti utasítás után a vonalra, hogy megvalósítása szerint mi jelenik meg a standard kimeneten az utasítás hatására! (3p)

4. Készítsen C++ nyelven a koordináták beolvasásához olyan függvényt, amivel azok a fenti formátumban megadva beolvashatók a megszokott módon egy istream típusú objektumról. Helyes bemeneti sorozat esetén működjön helyesen az alábbi kódrészlet, azaz fájl végéig olvasson: (4p)

```
Koord k;
while (std::cin >> k) { std::cout << k; }
```

3. Feladat megoldása:

```
std::ostream& operator<<(std::ostream& os, const Koord& k) {
    return os << k.x << ";" << k.y;
}
```

4. Feladat megoldása:

```
std::istream& operator>>(std::istream& is, Koord &k) {
    is >> k.x;
    return is.ignore(1) >> k.y; // igore helyett egy char olvasás is megfelel
}
```