rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2016.03.07. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

Az első két 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. Jelölje, hogy mely kijelentés(ek) igaz(ak) C++ nyelvben a kivételkezelésre! (2p)

   ☐ 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 **new** képes kivételt generálni.
   ☐ A **throw** értéket dob, ezért másoló konstruktort hívhat.

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

```
{ const int i = 12; int& r = i; void fv(int& j); fv(i);}
```

   ☐ `i` deklarációja hibás, mert konstans nem kaphat értéket.
   ☐ `i` nem adható át fv-nek, mert `i` konstans.
   ☐ `i` konstans, ezért csak konstans referencia hivatkozhat rá!
   ☐ A `void fv(int& j)` deklaráció hibás, mert a referenciát mindig inicializálni kell.
   ☐ `A void fv(int& j);` deklaráció rossz helyen van.
   ☐ A kódrészlet hibás, mert referenciát csak globális változóra lehet létrehozni.

3. **Tervezzen** egy olyan osztályt (Ido), ami idő (óra, perc) tárolására alkalmas! Az osztálynak (6p)
   - legyen paraméter nélkül hívható konstruktora, ami 0 óra 0 percet állít;
   - legyen olyan beállító függvénye (set), amivel az óra és a perc is beállítható; amennyiben csak az órát adják meg paraméterként, úgy adott óra 0 percet állítson be a függvény;
   - legyenek olyan lekérdező függvényei, amelyekkel az óra és a perc külön-külön lekérdezhető;
   - legyen olyan operátora (<), amivel el lehet dönteni, hogy két időpont közül melyik a korábbi;

   Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. A beállító függvény dobjon **const char *** hibát, ha hibás paraméterrel hívják (pl. negatív perc)!
   Működjön helyesen az alábbi kódrészlet:

```
Ido t0, t10, t13;
t10.set(10);                                  // 10 óra 00 percet állít be
t13.set(13,21);                               // 13 óra 21 percet állít be
std::cout << std::boolalpha << (t13 < t10);   // false-t ír ki
```

Egy lehetséges megoldás:

```
class Ido {
    int ora;
    int perc;
public:
    Ido() :ora(0), perc(0) {}
    void set(unsigned int o, unsigned int p = 0);
    int getOra() const { return ora; }
    int getPerc() const { return perc; }
    bool operator<(const Ido& rhs) const;
};

void Ido::set(unsigned int o, unsigned int p) {
    if (o > 23 || p > 59) throw "Hiba";
    ora = o;
    perc = p;
}

bool Ido::operator<(const Ido& rhs) const {
    return ora*60+perc < rhs.ora*60+rhs.perc;
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 1. ellenőrző dolgozat. 2016.03.07. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

Az első két 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. Jelölje, hogy mely kijelentés(ek) igaz(ak) a C++ nyelvre! (2p)

- ☐ A struktúra egy osztály.
- ☐ A **new** képes kivételt generálni.
- ☐ A nyelvben nincs logikai típus.
- ☐ A konstruktor mindig int típusú, ezért azt nem kell kiírni.
- ☐ Az értékadó operátor nem terhelhető túl.
- ☐ A névterek nem ágyazhatók egymásba.

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

```
{ int i = 12; int& r; int& fv(const int& j); fv(i);}
```

- ☐ `i` deklarációja hibás.
- ☐ Az `int& fv(const int& j);` deklaráció rossz helyen van.
- ☐ Nincs hiba a kódrészletben.
- ☐ `i` nem adható át fv-nek, mert `i` nem konstans.
- ☐ `int& r;` deklaráció hibás, mert a referenciát mindig inicializálni kell.
- ☐ Lehet referencia értékű függvény.

3. **Tervezzen** egy olyan osztályt (Pont), ami olyan síkbeli pontok koordinátáinak (X,Y) tárolására való, melyek az origó középpontú egységkörön belül esnek. Az osztálynak (6p)

- legyen paraméter nélkül hívható konstruktora, ami mindkét koordinátát nullára állítja;
- legyen olyan beállító függvénye (set), amivel mind az X, mind az Y koordináta beállítható; amennyiben csak egy paraméterrel hívják, csak az X értéket állítsa, az Y koordináta legyen 0;
- legyenek lekérdező függvényei, amelyekkel mindkét koordináta külön-külön lekérdezhető;
- legyen olyan operatora (<), amivel el lehet dönteni, hogy a két pont közül melyik van közelebb az origóhoz;

Valósítsa meg az osztályt C++ nyelven úgy, hogy az osztálynak legyen legalább 1 nem inline tagfüggvénye! Ne lehessen az adattagokhoz kívülről közvetlenül hozzáférni. A beállító függvény dobjon **const char *** hibát, ha a pont kívül esne az egységkörön!
Működjön helyesen az alábbi kódrészlet:

```
Pont p0, p10, p13;
p10.set(0.10);                                  // X = 0.1, Y = 0.0
p13.set(0.93,0.2);                              // X = 0.93 Y = 0.2
std::cout << std::boolalpha << (p13 < p10);  // false-t ír ki
```

Egy lehetséges megoldás:

```
class Pont {
    double x;
    double y;
public:
    Pont() :x(0), y(0) {}
    void set(double _x, double _y = 0);
    int getX() const { return x; }
    int getY() const { return y; }
    bool operator<(const Pont& rhs) const;
private:
    double dist2() const { return x * x + y * y; }
};

void Pont::set(double _x, double _y) {
    if (_x *_x + _y *_y > 1) throw "Hiba";
    x = _x;
    y = _y;
}
bool Pont::operator<(const Pont& rhs) const {
    return dist2() < rhs.dist2();
}
```