rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.23. Kurz/**Terem**: Gy1/	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)

Referencia típusú tagváltozó csak inicializáló listán inicializálható.	**I**	H
Absztrakt osztálynak nem kell, hogy legyen virtuális tagfüggvénye.	I	**H**
Minden osztályból létre lehet hozni tömböt.	I	**H**
Privát (**`private`**) adattagot a származtatott objektumból mindig el lehet érni közvetlenül.	I	**H**

2. A *Pont* osztály síkbeli pontok koordinátainak tárolására alkalmas. Az osztály felhasználásával, annak módosítása nélkül hozzon létre egy *Pont3D* osztályt, ami térbeli pont tárolására alkalmas! Az osztály adattagjaihoz közvetlenül ne lehessen hozzáférni! Készítsen *inserter* operátort is, amivel egy térbeli pont koordinátai kiírhatók egy *std::ostream* jellegű adatfolyamra! (6p)

```
class Pont {
    int x, y;
public:
    Pont(int x, int y) :x(x), y(y) { }
    int getx() const { return x; }
    int gety() const { return y; }
};
```

Működjön az elvárásoknak megfelelően az alábbi kódrészlet! (Figyeljen a konstruktorokra!)

```
{ Pont3D p0, p4(1,2,3);
  std::cout << p0 << std::endl;                          // kiír: (0,0,0)
  const Pont3D p1 = p4;
  std::cout << p1 << std::endl;                          // kiír: (1,2,3)
  std::cout << p1.getx() << p1.getz() << std::endl;      // kiír  13
}
```

```
class Pont3D : public Pont {
    int z;
public:
    Pont3D(int x = 0, int y = 0, int z = 0) :Pont(x,y), z(z) {}
    int getz() const { return z; }
};
```

```
Tartalmazott objektummal többet kellett írni:
class Pont3D {
    Pont p;
    int z;
public:
    Pont3D(int x = 0, int y = 0, int z = 0) :p(x,y), z(z) {}
    int getx() const { return p.getx(); }
    int gety() const { return p.gety(); }
    int getz() const { return z; }
};
```

```
std::ostream& operator<<(std::ostream& os, const Pont3D& p) {
   return os << '(' << p.getx() << ',' << p.gety() << ',' << p.getz() << ')';
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.23. Kurz/**Terem**: Gy1/	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 implicit másoló konstruktor nem hívja meg az alaposztály másoló konstruktorát.	I	**H**
Referencia típusú tagváltozót konstruktor törzsében lehet inicializálni.	I	**H**
A throw utasítás értéket dob, ezért másoló konstruktort hívhat.	I	H
Privát (**`private`**) adattagot a származtatott objektumból közvetlenül nem lehet elérni.	I	H

2. A Vektor osztály kétdimenziós síkbeli pontok koordinátainak tárolására alkalmas. Az osztály felhasználásával, annak módosítása nélkül hozzon létre egy Vektor3D osztályt, ami térbeli vektor tárolására alkalmas! Az osztály adattagjaihoz közvetlenül ne lehessen hozzáférni! Készítsen *inserter* operátort is, amivel egy térbeli vektor koordinátai kiírhatók egy *std::ostream* jellegű adatfolyamra! (6p)

```
class Vektor {
    double v[2];
public:
    Vektor(double x, double y) { v[0] = x;, v[1] = y; }
    double getx() const { return v[0]; }
    double gety() const { return v[1]; }
};
```

Működjön az elvárásoknak megfelelően az alábbi kódrészlet! (Figyeljen a konstruktorokra!)

```
{ Vektor3D v0, v4(1,2,3);
  std::cout << v0 << std::endl;                              // kiír: (1,1,1)
  const Vektor3D v1 = v4;
  std::cout << v1 << std::endl;                              // kiír: (1,2,3)
  std::cout << v1.getx() << v1.getz() << std::endl;          // kiír  15
}
```

```
class Vektor3D : public Vektor {
    double z;
public:
    Vektor3D(double x = 1, double y = 1, double z = 1) :Vektor(x,y), z(z) {}
    double getz() const { return z; }
};
```

```
Tartalmazott objektummal többet kellett írni:
class Vektor3D {
    Vektor v;
    double z;
public:
    Vektor3D(int x = 1, int y = 1, int z = 1) :v(x,y), z(z) {}
    int getx() const { return v.getx(); }
    int gety() const { return v.gety(); }
    int getz() const { return z; }
};
```

```
std::ostream& operator<<(std::ostream& os, const Vektor3D& v) {
   return os << '(' << v.getx() << ',' << v.gety() << ',' << v.getz() << ')';
}
```