rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.24. Kurz/**Terem**: Gy4/	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)

Tartalmazott objektum destruktora csak akkor fut le, ha az virtualis	I	**H**
Bármelyik tagfüggvény lehet virtuális.	I	**H**
Az implicit konstruktor meghívja tartalmazott objektumok paraméter nélkül hívható konstruktorát.	**I**	H
A { class Obj {}; Obj *o = new Obj(Obj()); } kódrészlet végrehatásakor Obj értékadó operátora is hívódik.	I	**H**

2. A *CalendarEntry* osztály naptárbejegyzéseket tárol. Minden bejegyzés egy leírásból (*String*) áll. A *CalendarEntry* osztály felhasználásával, annak módosítása nélkül hozzon létre egy *CalendarEntryPlace* osztályt, ami mindenütt használható, ahol a *CalendarEntry*, de konstruktorparaméterként egy helyszínt (*String*) is meg lehet adni, és kiíratáskor az így megadott helyszínt is kiírja! (6p)

```
class CalendarEntry {
   String desc;        // leírás
public:
   CalendarEntry(String s) :desc(s) {}
   virtual void print() const { std::cout << desc; }
   virtual ~CalendarEntry() {}
};
```

A *CalendarEntryPlace* osztályt úgy valósítsa meg, hogy adattagjai közvetlenül ne legyenek elérhetőek, és az alábbi kódrészlet az elvárásoknak megfelelően működjön! A megvalósításnál ügyeljen a konstans objektumokra és a végtelen ciklus elkerülésére!

```
const CalendarEntry meeting("Meeting");
CalendarEntryPlace randi("Randi", "Buliban");      // helyszín: buliban
CalendarEntryPlace randi2(meeting);                // a helyszín ilyenkor: ""
const CalendarEntry* e[3] = { &meeting, &randi, &randi2 };
for (int i = 0; i < 3; i++) {
    e[i]->print(); std::cout << std::endl;
}                                    // kiír: Meeting\nRandi @Buliban\nMeeting @\n
```

```
// Egy lehetséges megoldás
class CalendarEntryPlace : public CalendarEntry {
   String place;
public:
   CalendarEntryPlace(String s, String p) :CalendarEntry(s), place(p) {}
   CalendarEntryPlace(const CalendarEntry& ce) :CalendarEntry(ce), place("") {}
   void print() const {
       CalendarEntry::print(); std::cout << " @" << place;
    }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 4. beszámoló dolgozat. 2017.03.24. Kurz/**Terem**: Gy4/	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)

Tartalmazott objektum destruktora mindig lefut, ha a tartalmazó destruktora lefut.	**I**	H
Nem lehet bármelyik tagfüggvény virtuális.	**I**	H
Az implicit konstruktor nem hívja meg a tartalmazott objektumok paraméter nélkül hívható konstruktorát.	I	**H**
A { class Obj {}; Obj *o = new Obj(Obj()); } kódrészlet végrehatásakor Obj értékadó operátora nem hívódik.	**I**	H

2. A *NaptarBejegyzes* osztály naptárbejegyzéseket tárol. Minden bejegyzés leírásból (*String*) áll. A *NaptarBejegyzes* osztály felhasználásával, annak módosítása nélkül hozzon létre egy *NaptarBejegyzesHellyel* osztályt, ami kompatibilis a *NaptarBejegyzes* osztállyal, de konstruktorparaméterként egy helyszínt (*String*) is meg lehet adni, amit kiíráskor ki is ír! (6p)

```
class NaptarBejegyzes {
   String leiras;      // leírás
public:
   NaptarBejegyzes(String miez) :leiras(miez) {}
   virtual void kiir() const { std::cout << leiras; }
   virtual ~NaptarBejegyzes(){}
};
```

A *NaptarBejegyzesHellyel* osztályt úgy valósítsa meg, hogy adattagjai közvetlenül ne legyenek elérhetőek, és az alábbi kódrészlet az elvárásoknak megfelelően működjön! A megvalósításnál ügyeljen a konstans objektumokra és a végtelen ciklus elkerülésére!

```
const NaptarBejegyzes megbesz("Fonokkel");
NaptarBejegyzesHellyel randi("Randi", "Mozi elott");     // helyszín: Mozi elott
NaptarBejegyzesHellyel marmegint(megbesz);               // helyszín ilyenkor: "*"
const NaptarBejegyzes* e[3] = { &megbesz, &randi, &marmegint };
  for (int i = 0; i < 3; i++) {
    e[i]->kiir(); std::cout << std::endl;
  }               // Kiír: Fonokkel\nRandi hol:Mozi elott\nFonokkel hol:*\n
```

```
// Egy lehetséges megoldás
class NaptarBejegyzesHellyel : public NaptarBejegyzes {
   String hely;
public:
   NaptarBejegyzesHellyel(String s, String h)
                                          :NaptarBejegyzes(s), hely(h) {}
   NaptarBejegyzesHellyel(const NaptarBejegyzes& nb)
                                          :NaptarBejegyzes(nb), hely("*") {}
   void kiir() const {
       NaptarBejegyzes::kiir(); std::cout << " hol:" << hely;
   }
};
```