AAAAAAAAAAAAAAAAAA rendelkezésre álló idő: **15 perc**

Szoftver laboratórium 2.	4. Ellenőrző dolgozat.	2014.04.15.	Kurz/**Terem**: L4/
Név:	**Neptun:**		Összpont:

Tételezze fel, hogy rendelkezésére áll az előző laboron elkészített *Tartozek* osztály:

```
class Tartozek {
   String gySzam;
   String megnevezes;
public:
   Tartozek(const char*, const char*);
   virtual void print(std::ostream&) const;
   virtual ~Tartozek();
};
```

---

1. feladat 2 pont

A *Tartozek* osztály felhasználásával, annak módosítása nélkül hozzon létre egy olyan *Lapolvaso* osztályt, ami együtt tárolható a már meglevő tartozékokkal! Az új osztály tárolja a lapolvasó gyártójának nevét (*String*)! Valósítsa meg az új osztály minden szükséges tagfüggvényét úgy, hogy a *print* tagfüggvény a gyártó nevét is írja ki a lapolvasó egyéb jellemzőivel (gyári szám, megnevezés) együtt!

---

2. feladat 3 pont

Tételezze fel, hogy rendelkezésére áll egy generikus tömb (*Array*<*T*>), ami maximum 100 generikus adatot tud tárolni, és pontosan úgy viselkedik, mint a C nyelvben megismert tömb típus! (A tömb indexelhető. Az indexoperátorral kiválasztott elem értékadás bal és jobboldalán egyaránt állhat.) Hozzon létre a generikus tömb felhasználásával egy *Raktar* osztályt, ami alkalmas számítógép tartozékokat (*Printer, Monitor, Lapolvaso,* …) heterogén gyűjteményként tárolni! Legyen a *Raktar* osztálynak *add*, és *list* művelete, amivel új tartozékot tehetünk a raktárba, illetve ki tudjuk listázni annak tartalmát a szabványos kimenetre! A raktár adatait közvetlenül ne lehessen elérni! Ne legyen másolható, és az értékadás művelete se legyen elérhető! Feltételezheti, hogy raktárba maximum 100 tartozékot teszünk.

---

3. feladat 1 pont

Írjon programrészletet, ami betesz 2 db lapolvasót a raktárba, és kiírja azok adatait a szabványos kimenetre! Ügyeljen arra, hogy a kódrészletben ne legyen memóriaszivárgás!

```
class Lapolvaso : public Tartozek {
   String gyarto;
public:
   Lapolvaso(const char* gysz, const char* megn, const char* gyarto)
      : Tartozek(gysz, megn), gyarto(gyarto) {}
   void print(std::ostream& os) const {
      Tartozek::print(os);
      os << " gyarto: " << gyarto;
   }
};

class Raktar {
   Array<Tartozek*> tar;
   size_t db;
   Raktar(const Raktar&);
   Raktar& operator=(const Raktar&);
public:
   Raktar() :db(0) {}
   void add(Tartozek* t) { tar[db++] = t; }
   void list() {
      for(size_t i = 0; i < db; i++) tar[i]->print(std::cout);
   }
   ~Raktar() {
      for(size_t i = 0; i < db; i++) delete tar[i];
   }
};

{
   Raktar r;
   r.add(new Lapolvaso("1234", "Lapolvaso", "epson"));
   r.add(new Lapolvaso("5678", "Lapolvaso", "HP"));
   r.list();
}
```