rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 5. beszámoló dolgozat. 2017.04.20. Kurz/**Terem**: Gy2/	Elért pontszám:
Név: **Neptun:**	

1. Készítsen **absztrakt** alaposztályt kávéfajták (*Coffee*) heterogén kollekcióban történő tárolására! Az osztálynak ne legyen adattagja, csak egy *void print()* metódusa, ami képes a szabványos kimenetre kiírni a konkrét kávéfaja adatait! Az osztály felhasználásával hozzon létre egy konkrét kávéfajtát (*Java*), ami kiírja a konstruktorában kapott szöveget. A szöveg tárolására használja az *std::string* osztályt! (4p)

```
struct Coffee {
   virtual void print() = 0;
   virtual ~Coffee() {}
};

class Java: public Coffee {
   std::string txt;
public:
   Java(const char *txt) :txt(txt) {}
   void print() { std::cout << txt; }
};
```

2. Az *std::deque* tároló felhasználásával hozzon létre egy termoszt (*Thermos*), ami képes az előző feladatban megfogalmazott kávéfajtákat tárolni! Legyen a *Thermos* osztálynak egy *fill()* metódusa, amivel kávét tölthetünk a termoszba. Egyszerre több kávéfajta is lehet a termoszban. A termosz tartalma áttölthető egy másik termoszba az *operator=* művelettel. Ekkor az eredeti termosz kiürül, tartalma hozzáadódik a cél termosz tartalmához, miközben kiírja a szabványos kimenetre az eredeti termoszban levő kávék adatait. Feltételezhető, hogy saját magába nem lehet áttölteni. Ha a termosz eltörik (megsemmisül), akkor a benne levő kávé is megsemmisül. A *Thermos* osztály adatait közvetlenül ne lehessen elérni! A felsoroltakon és az implicit módon is keletkező tagfüggvényeken kívül más publikus függvénye ne legyen! Az alábbi kódrészlet a használatra mutat példát: (6p)

```
Thermos t1, t2;
t1.fill(new Java("arabica"));
t1.fill(new Java("robusta"));
t1.fill(new Irish(30));      // az Irish osztályt nem kell megvalósítania!
t2 = t1;
```

std::deque fontosabb tagjai: push_back(), pop_back(), push_front(), pop_front(), front(), back(), size(), capacity(), reserve(), operator[], at(), insert(), erase(), iterator

```
class Thermos {
    std::deque<Coffee*> t;
public:
    Thermos& operator=(Thermos& rhs) {
        for (size_t i = 0; i < rhs.t.size(); i++) {
            rhs.t[i]->print();
            t.push_back(rhs.t[i]);
        }
        rhs.t.clear();
        return *this;
    }
    void fill(Coffee *coffee) { t.push_back(coffee); }
    ~Thermos() {
    for (size_t i = 0; i < t.size(); i++)
       delete t[i];
    }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapjai 2. 5. beszámoló dolgozat. 2017.04.20. Kurz/**Terem**: Gy2/	Elért pontszám:
Név: **Neptun:**	

1. Készítsen **absztrakt** alaposztályt kávéfajták (*Kave*) heterogén kollekcióban történő tárolására! Az osztálynak ne legyen adattagja, csak egy *void kiir()* metódusa, ami képes a szabványos kimenetre kiírni a konkrét kávéfaja adatait! Az osztály felhasználásával hozzon létre egy konkrét kávéfajtát (*Presszo*), ami kiírja a konstruktorában kapott szöveget. A szöveg tárolására használja az *std::string* osztályt! (4p)

```
struct Kave {
   virtual void kiir() = 0;
   virtual ~Kave() {}
};

class Presszo: public Kave {
   std::string txt;
public:
   Presszo(const char *txt) :txt(txt) {}
   void kiir() { std::cout << txt; }
};
```

2. Az *std::deque* tároló felhasználásával hozzon létre egy termoszt (*Termosz*), ami képes az előző feladatban megfogalmazott kávéfajtákat tárolni! Legyen a *Termosz* osztálynak egy *betolt()* metódusa, amivel kávét tölthetünk a termoszba. Egyszerre több kávéfajta is lehet a termoszban. A termosz tartalma áttölthető egy másik termoszba az *operator*= művelettel. Ekkor az eredeti termosz kiürül, tartalma hozzáadódik a cél termosz tartalmához, miközben kiírja a szabványos kimenetre az eredeti termoszban levő kávék adatait. Feltételezhető, hogy saját magába nem lehet áttölteni. Ha a termosz eltörik (megsemmisül), akkor a benne levő kávé is megsemmisül. A Termosz osztály adatait közvetlenül ne lehessen elérni! A felsoroltakon és az implicit módon is keletkező tagfüggvényeken kívül más publikus függvénye ne legyen! Az alábbi kódrészlet a használatra mutat példát: (6p)

```
Termosz t1, t2;
t1.betolt(new Presszo("arabica"));
t1.betolt(new Presszo("robusta"));
t1.betolt(new Irish(80));     // az Irish osztályt nem kell megvalósítania!
t2 = t1;
```

std::deque fontosabb tagjai: push_back(), pop_back(), push_front(), pop_front(), front(), back(), size(), capacity(), reserve(), operator[], at(), insert(), erase(), iterator

```
class Termosz {
    std::deque<Kave*> t;
public:
    Termosz& operator=(Termosz& rhs) {
        for (size_t i = 0; i < rhs.t.size(); i++) {
            rhs.t[i]->kiir();
            t.push_back(rhs.t[i]);
        }
        rhs.t.clear();
        return *this;
    }
    void betolt(Kave* coffee) { t.push_back(coffee); }
    ~Termosz() {
    for (size_t i = 0; i < t.size(); i++)
       delete t[i];
    }
};
```