AAAAAAAAAAAAAAAAAAAAAAAAAAAA

Programozás alapjai II. 5. ellenőrző dolgozat. 2013.05.02. Kurz/**Terem**: G1/	**20 perc**
Név: **Neptun:**	Összpont:

1. feladat 3 pont

Tételezze fel, hogy rendelkezésére áll egy verem tulajdonságokkal rendelkező generikus tároló (*Stack*)! A tároló a szokásos verem műveletekkel rendelkezik (*push, pop, top, empty*), és csak default konstruktora van. Ezen osztály **felhasználásával** készítsen egy **véges méretű** generikus vermet (*MyStack)*! A maximális méretet a konstruktorban lehessen megadni, amelynek alapértelmezett értéke legyen 100! A *MyStack* osztály a *Stack* osztállyal **teljesen azonos** módon viselkedjen, kivéve akkor, amikor eléri a maximális méretet. Ekkor dobjon *std::out_of_range()* hibát!

Örökléssel egyszerűbb, de tartalmazott objektummal is megoldható volt.
Ez utóbbi esetben a Stack osztály minden függvényét delegálni kellett.

```
template<typename T>
class MyStack :public Stack<T> {
   size_t maxsiz;
public:
   MyStack(size_t n = 100) : maxsiz(n) {}
   void push(const T& a) {
      if (Stack<T>::size() >= maxsiz) throw std::out_of_range("MyStack: push");
      Stack<T>::push(a);
   }
};
```

2. feladat 3 pont

Írjon függvénysablont (*myunique*), ami az *std::unique* sablonhoz hasonlóan elhagyja egy sorozatból az egymás után ismétlődőket! A függvény első két paramétere két iterátor (ForwardIterator), ami kijelöli a jobbról nyílt intervallum kezdetét és végét, a 3. paramétere egy predikátum függvény (kétparaméteres), amivel ez elemek egyezősége tesztelhető. A függvény visszatérési értéke egy iterátor, ami a módosított sorozat végét jelöli ki. Amennyiben helyesen oldja meg a feladatot, akkor az alábbi kódrészlet a következőt írja ki: **1,2,3,17,3**

```
using namespace std;
int v[] = { 1, 2, 2, 3, 3, 17, 3 };
ostream_iterator<int> out(cout, ",");
copy(v, myunique(v, v+7, std::equal_to<int>()), out);
```

```
template <class Iter, class Func>
Iter myunique(Iter first, Iter last, Func func) {
    if (first == last)
        return last;
    Iter out = first;
    while (++first != last)
        if (!func(*out, *first))
            *(++out) = *first;
    return ++out;
}
```