AAAAAAAAAAAAAAAAA rendelkezésre álló idő: **15 perc**

Programozás alapjai 2.	5. ellenőrző dolgozat.	2014.05.07.	Kurz/**Terem**: G2/	
Név:		**Neptun:**		Összpont:

1. feladat 3 pont

Írjon függvénysablont (***monoton_e***), ami megállapítja, hogy egy adatsorozat monoton sorozatot képez-e! Az elemek összehasonlításához a 3. paramétereként megadott kétparaméteres függvényt/függvényobjektumot használja! A *monoton_e* függvény első két paramétere két iterátor, ami kijelöli az adatsorozat kezdetét és végét (jobbról nyílt intervallum). Ha jól oldja meg a feladatot, akkor az alábbi kódrészlet lefutása után az *eredmeny* változó igaz értéked vesz fel. (A kettőnél kevesebb elemet tartalmazó sorozatot monotonnak tekintjük.)

```
    inline bool hasonlit(int a, int b) { return a < b; }
    int sorozat[] = { 1, 4, 9, 16, 25};     // a sorozat
...
    bool eredmeny = monoton_e(sorozat, sorozat+5, hasonlit);
```

```
template<typename Iterator, typename Predicate>
bool monoton_e(Iterator first, Iterator last, Predicate cmp) {
  if (first == last)
    return true;
  Iterator prev = first++;
  while (first != last)
    if (!cmp(*prev++, *first++))
      return false;
  return true;
}
```

---

2. 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 paraméter nélkül hívható 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!

```
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);
   }
};
```