AAAAAAAAAAAAAAAAA rendelkezésre álló idő: **15 perc**

Programozás alapjai 2.	5. ellenőrző dolgozat.	2014.05.08.	Kurz/**Terem**: G1/	
Név:		**Neptun:**		Összpont:

1. feladat 3 pont

Írjon függvénysablont (***mycopy_if***), ami egy adatsorozat elemeit akkor másolja át egy másik tárolóba, ha azok egy adott feltételnek eleget tesznek (predikátum)! A *mycopy_if* függvény első három paramétere három iterátor, amelyek közül az első kettő a bemeneti adatokat jelöli ki (jobbról nyílt intervallum kezdete és vége). A harmadik iterátor pedig az eredményt tároló sorozat elejére mutat. A függvény negyedik paramétere a predikátum függvény/függvényobjektum legyen! A függvény visszatérési értéke egy olyan iterátor érték legyen, ami az eredménysorozat utolsó eleme után mutat! Ha jól oldja meg a feladatot, akkor az alábbi kódrészlet lefutása után az out tömb tartalma 1,3,5 lesz, az ende pointer pedig out+3 értéket vesz fel.

```
    inline bool odd(int i) { return i&1; }
    double inp[] = { 1, 2, 3, 4, 5};      // bemeneti sorozat
    double out[5];                        // eredmény helye
...
    double *ende = mycopy_if(inp, inp+5, out, odd);
```

```
template<class Iterator, class Predicate>
Iterator mycopy_if(Iterator first, Iterator last, Iterator result, Predicate pred) {
  while (first!=last) {
    if (pred(*first))
      *result++ = *first;
    first++;
  }
  return result;
}
```

---

2. feladat 3 pont

Tételezze fel, hogy rendelkezésére áll egy sorozattároló (*Vector*)! A tároló a szokásos vektorműveletek egy részhalmazával rendelkezik (*push_back, pop_back, front, back, size, resize*), és csak paraméter nélkül hívható konstruktora van. Ezen osztály **felhasználásával készítsen** egy véges méretű generikus tárolót (*MyVector)*! A maximális méretet a konstruktorban lehessen megadni, amelynek alapértelmezett értéke legyen 350! A *MyVector* osztály a *Vector* osztállyal teljesen azonos módon viselkedjen, kivéve akkor, amikor eléri a maximális méretet. Ekkor a legutolsó elem (*back*) helyére tegye az új elemet! Ha a tárolt elemek száma csökken, akkor ismét a megszokott módon működjön!

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

```
template<typename T>
class MyVector :public Vector<T> {
   size_t maxsiz;
public:
    MyVector(size_t n = 350) : maxsiz(n) {}
    void push_back(const T& a) {
      if (Vector<T>::size() >= maxsiz) Vector<T>::back() = a;
      else Vector<T>::push_back(a);
    }
// a teljes megoldáshoz a resize() is hozzátartozik, de elfogadtuk enélkül is
    void resize(size_t n, const T& a = T()) {
      if (n > maxsiz) Vector<T>::resize(maxsiz, a);
   }
};
```