AAAAAAAAAAAAAAAAAAAAAA

Szoftver laboratórium II. 5. ellenőrző dolgozat. 2013.05.07. Kurz/**Terem**: L2/	**15 perc**
Név: **Neptun:**	Összpont:

1. feladat **4 pont**

A *Tarolo* sablon egy egyszerű sorozattárolót valósít meg, melynek az automatikusan létrejövő konstruktorai mellett a következő publikus tagfüggvényei vannak:

Tagfüggvény	**Funkció**
`void push_back(const T&);`	hozzáfűz egy elemet a tárolt sorozat végéhez
`void pop_back();`	kitörli a legutolsó tárolt adatot
`T& back();`	visszaadja a tárolt sorozat utolsó elemének referenciáját
`const T& back() const;`	

Hozzon létre egy *MyStack* adaptert, amivel a *Tarolo* sablonból vermet lehet készíteni. Valósítsa meg a szokásos veremműveleteket (*push, pop, top, size, empty*)! A *MyStack* adapterrel létrehozott objektumok legyenek átadhatók paraméterként és szerepelhessenek értékadás jobb ill. bal oldalán is! **(3.5p)**

Milyen követelményeket kell támasztani a generikus adattal szemben, hogy minden fenti előírást meg tudjon valósítani? (0.5p)

```
template <class T, class Container = Tarolo<T> >
class MyStack : private Tarolo<T> {
    size_t siz;
public:
    MyStack() : siz(0) {}
    void push(const T& t) {
        ++siz;
        push_back(t);
    }
    void pop() {
        --siz;
        pop_back();
    }
    T& top() { return back();
    }
    size_t size() const { return siz; }
    bool empty() const { return siz == 0; }
};
```

2. feladat **2 pont**

Készítsen függvénysablont, ami a paraméterként kapott verem elemeinek sorendjét megfordítja. Az eredmény az eredetileg átadott adat helyén keletkezzen! Mutassa be a függvénysablon használatát!

```
template <class T>
void reverse(T& st) {
    T tmp;
    while (!st.empty()) {
        tmp.push(st.top());
        st.pop();
    }
    st = tmp;
}

MyStack<int> st;
reverse(st);
```