AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő: 18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2015.04.24. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. Írjon C++ függvénysablont (*cserel*), ami felcseréli a paraméterként kapott két generikus adatot! Ügyeljen arra, hogy a csere ténylegesen megtörténjen, ne csak a formális paramétereken menjen végbe! 2p
2. Az 1. feladatban elkészített *cserel* sablon felhasználásával készítsen C++ függvénysablont (*fordít*), ami egy iterátorokkal megadott adatsorozat elemeinek sorrendjét cserékkel megfordítja, azaz az első elemet felcseréli az utolsóval, a második elemet az utolsó előttivel, stb! A függvény paraméterként kapja az adatsorozatot kijelölő iterátorokat (jobbról nyílt intervallum). Mindkét iterátor ún. kétirányú iterátor, azaz a növelés, és a csökkentés műveletei is értelmezettek. 4p
3. A 2. feladatban elkészített *fordit* sablon felhasználásával írjon C++kódrészletet, melyben a standard inputról beolvas egy szót és eldönti, hogy az palindrom-e, azaz visszafelé olvasva megegyezik önmagával (pl. sörös, tejet, lehel, ...)! A szó beolvasásához és tárolásához használja az *std::string* tárolót, vagy a laborgyakorlatokon elkészített *String* osztályt! Tételezze fel, hogy mindkettőnek van kétirányú iterátora! 3p
4. Hozzon létre egy 10 elemű C tömböt! Mutassa meg, hogy a *fordit* sablon felhasználásával hogyan lehet az elemek sorrendjét megfordítani a tömbben! 1p

Egy lehetséges megoldás:

```
template <class T>
void cserel(T& a, T&b) {
    T t(a); a = b; b = t;
}

template <class Iter>
void fordit(Iter first, Iter last) {
    while (first != last && first != --last) {
        cserel(*first, *last);
        ++first;
    }
}

std::string szo1, szo2
std::cin >> szo1;
szo2 = szo1;
fordit(szo1.begin(), szo1.end());
if (szo1 == szo2) std::cout << "palindrom" << std::endl;

int t[10];
fordit(t, t+10);
```

BBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2015.04.24. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. Írjon C++ függvénysablont (*swap*), ami felcseréli a paraméterként kapott két generikus adatot! Ügyeljen arra, hogy a csere ténylegesen megtörténjen, ne csak a formális paramétereken menjen végbe! 2p
2. Az 1. feladatban elkészített *swap* sablon felhasználásával készítsen C++ függvénysablont (*sullyeszt*), ami egy iterátorokkal megadott generikus adatsorozat legnagyobb elemét cserékkel az adatsorozat végére viszi, azaz végrehajtja a buborék rendezés egy belső ciklusát! A függvény paraméterként kapja az adatsorozatot kijelölő iterátorokat (jobbról nyílt intervallum). Tételezze fel, hogy a generikus adaton értelmezett a > (nagyobb, mint) operátor! Ha jól oldja meg a feladatot, akkor az alábbi kódrészlet lefutása után a zh tömb tartalma `{ 3,8,2,1,9 }` lesz. 4p

```
int zh[] = { 8, 3, 9, 2, 1 };
sullyeszt(zh, zh+5);
```

3. Hozzon létre az *std::vector* sablon felhasználásával egy vektort, amiben a fenti példa *zh* tömbjének adatai vannak! A 2. feladatban elkészített *sullyeszt* sablon, valamint az *std::vector* iterátoros interfészének felhasználásával rendezze a vektor elemeit buborékrendezéssel! (A vektor iterátora használható kétirányú iterátorként.) 4p

Egy lehetséges megoldás:

```
template <class T>
void swap(T& a, T&b) {
    T t(a); a = b; b = t;
}

template <class Iter>
void sullyeszt(Iter first, Iter last) {
    if (first == last) return;
    Iter prev = first++;
    while (first != last) {
        if (*prev > *first) swap(*prev, *first);
        prev = first++;
    }
}

std::vector<int> vc(zh, zh+5);
std::vector<int>::iterator i1=vc.begin(), i2=vc.end();
while (i1 != i2)
    sullyeszt(i1, i2--);
```