65. AA Megoldó Péter AA 65.

Programozás alapjai 2. (inf.) pótzárthelyi 2006.05.25. gyakolat: **G3/IE.217**	Érdemjegy:
terem:E.I.B/65.	Hftest:

Minden beadandó lap tetejére írja fel balra a feladat számát, jobbra a nevét és tankörét! A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll.
A feladatok megoldásához csak a letölthető C és C++ összefoglaló, valamint egy A4-es lapra saját kézzel írt „puska" használható. Számítógép, notebook, menedzser kalkulátor, rádiótelefon nem használható.

A feladatokat figyelmesen olvassa el! Ne írjon felesleges kódot, a feladatra koncentráljon! Jó munkát!

F.	Max.	Elért.
1	5	
2	2	
3	3	
4	3	
5	7	
Σ	**20**	

1. Feladat 5 pont

Írjon generikus dinamikus string osztályt, amely:

- képes elemeket beszúrni (insert)! (Az elemek számát tárolja, mivel általános (generikus) esetben nem tudjuk megmondani, mi jelzi a string végét!)
- képes két stringet egymás után másolni (concat)!
- képes visszaadni egy adott indexű elemet a szokásos index operátorral ([ ])! Ha az index érvénytelen, dobjon kivételt (felhasználhatja az STL kivételosztályát)!
- átadható érték szerint függvényparaméterként!
- helyesen kezeli a többszörös értékadást (*s1=s2=s3,* ahol *s1*, *s2*, s*3* ugyanazzal a típussal példányosított template-ek)!
- Milyen követelményeket támaszt az osztály a tárolandó típusokkal szemben (pl. alapértelmezett konstruktor rendelkezésre állása)?

Írjon egy egyszerű, maximum 10 utasításból álló programrészletet, amely bemutatja az osztály szolgáltatásainak használatát char típusra. Mutassa be a kivételkezelést is!

2. Feladat 2 pont

Döntse el, hogy hibásak-e az alábbi programrészletek! Ha igen, adja meg a hiba **helyét** és magyarázatát!
A programrészletekben elírások nincsenek, az összes meghívott függvény ill. hivatkozott osztály létezik, ha nincs megadva a kódrészletben, akkor feltételezze a helyes működést. A memóriaszivárgás is hibának minősül! Válaszait a kódrészletek mellett ill. alatt adja meg!

a)
```
Stq::Stq (const Stq& q){ ... Stq a = q; ...}
```
Hiba oka, ha van:
Másoló konstruktor önmagát hívja!

b)
```
class A { ... explicit A(int i65) {...}; };
class B { ...          B(int j65) {...}; };
...
A a = 65;
B b = 65;
```
Hiba oka, ha van:
A kontruktora explicit, ami tiltja az automatikus konverziót. (B esetében a B b = 65; -ből B b = B(65); lesz)

3. Feladat 3 pont

Tervezzen **komplex** számot reprezentáló osztályt, amely támogatja:

- a szorzás (*) műveletet a komplex számok között, a komplex és valós számok között, illetve a valós és a komplex számok között, valamint
- képes egy *ostream* típusú objektumra (pl: *cout*) kiírni önmagát!

Csak az osztály és az esetleges globális függvények **deklarációja** szükséges. A függvények **definíciója nem** része a feladatnak

```
class Komplex {
    double re, im;
public:
    Komplex();
    Komplex operator*(Komplex&);
    Komplex operator*(double);
    friend Komplex operator*(double, Komplex&);
    friend ostream& operator<<(ostream&, Komplex&);
};
```

4. Feladat 3 pont

Mit ír ki a szabványos kimenetre a program? Válaszához használja a négyzetrácsos területet!

```
#include <iostream>
using namespace std;
class A {
public:
    A(const int i = 0)   { cout << 'k' << i; }
    A(const A& a)        { cout << "c_q"; }
    virtual void f(A& a) { cout << 'f'; }
    ~A()                 { cout << 'd'; }
};
class B :public A {
    const char *pr;
public:
    B(const char *n = "q") :pr(n), A(65) { cout << 'K' << pr; }
    B(const B& b)        { cout << 'C'; }
    void f(A& a)         { cout << 'F'; }
    ~B()                 { cout << 'D'; }
};
int main() {
    B b("1ar");          cout << '\n';
    A a = b;             cout << '\n';
    A* p65 = new B;      cout << '\n';
    (*p65).f(a);         cout << '\n';
    delete p65;          cout << '\n';
    return(0);
}
```

k	6	5	K	1	a	r								
c	_	q												
k	6	5	K	q										
F														
d														
d	D	d												

5. Feladat 7 pont

Egy mosógép programját objektumorientáltan tervezzük meg, hogy minél könnyebben bővíthető legyen. Minden műveletet (***fűt***, ***mos***, ***öblít***, ***centrifugáz, stb***) egy-egy objektum valósít meg.
A műveletek végrehajtását egy *vezérlő* objektum vezérli. Ez maximum 200 műveletet tud **tárolni**, és végtelen ciklusban egymás után (a legutolsó után az első következik) **végrehajtani**. Az egyes műveletek végrehajtása az egyes objektumok ***Do*()** metódusának meghívásával történik. A vezérlő a *start()* metódussal indul és akkor áll meg, ha a végrehajtott művelet *Do()* metódusa hamis értékkel tér vissza. A mosási program előállítása a vezérlő objektum megfelelő feltöltéséből áll.

- **Tervezze** meg és vázolja fel az OO modell osztálydiagramját! Ügyeljen az osztályok felelősségének és hatáskörének helyes kialakítására! Egy új művelet-típus esetleges felvételéhez ne kelljen a már meglévő osztályokat módosítani!
- **Tervezze** meg a vezérlő osztály interfészét (publikus metódusait) úgy, hogy a mosási program könnyen előállítható, módosítható legyen, azaz lehessen új műveletet beszúrni, törölni, ill. a már letárolt műveletek számát lekérdezni.
- **Deklarálja** a *mos, öblít és vezérlő* objektumokat, illetve az alapobjektumot!
- **Definiálja (implementálja) a** vezérlő objektumot! Ügyeljen arra, hogy nincs mindig 200 művelete egy programnak!
- **Írjon** főprogramot, ami előállít egy 3 műveletes mosási programot és azt végrehajtja!

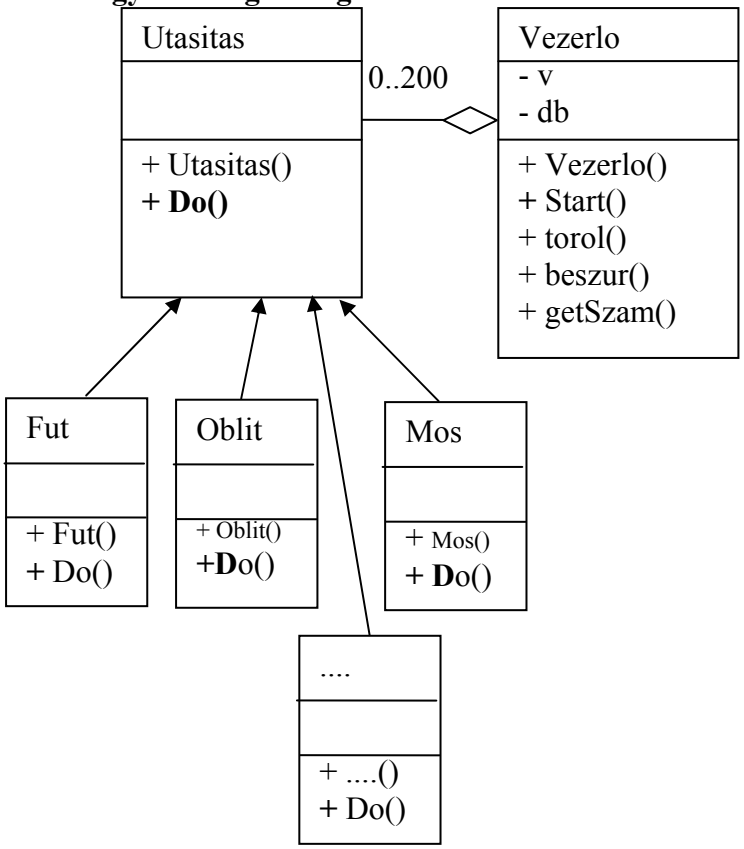
1. feladat egy lehetséges megoldása:

```
#ifndef FELADAT1_A
#define FELADAT1_A
// feladat1_a.hpp – generikus string sablonja

template <class T>
class String {
      T *tp;                          // Elemek tömbjére mutató pointer
      int nElem;                      // elemek száma
public:
    String() :tp(0), nElem(0){}
      String(const String&);          // copy konstruktor
    String concat(const String&);
      void insert(int, const T&);
      const T& operator[](int);       // csak jobb oldalon lehet
      String& operator=(const String&);
      ~String() { delete[] tp; }       // din. terület felszabadítása
};

template<class T>
String<T>::String(const String& e) {// copy konstruktor
    nElem = 0;                            // operator= híváshoz előkészítjük
    tp = 0;
    *this = e;                              // most már mehet
}

template<class T>
String<T> String<T>::concat(const String& e) {
    String tmp;
    tmp.tp = new T[tmp.nElem = nElem + e.nElem];
    int s = 0;
    for(int i = 0; i < nElem; i++) {
       tmp.tp[s++] = tp[i];
    }
    for(int i = 0; i < e.nElem; i++) {
       tmp.tp[i++] = e.tp[i];
    }
    return tmp;
}
```

```
template <class T>
void String<T>::insert(int ix, const T& e) {
    if (ix < 0) throw range_error("Index");
    int s = ix >= nElem ? ix : nElem;
    T *tmp = new T[s+1];
    int j = 0;
    for(int i = 0; i < nElem; i) {
       if (j == ix) j++;
       tmp[j++] = tp[i++];
    }
    tmp[ix] = e;
    delete tp;
    tp = tmp;
    nElem = s+1;
}

template<class T>
const T& String<T>::operator[](int ix) {
    if (ix < 0 || ix >= nElem) throw range_error("Index");
      return tp[ix];
}

template<class T>
String<T>& String<T>::operator=(const String& e) {    // értékadás operátor
      if(this != &e) {                    // nem önmagának ad értéket, akkor átmásol
        delete[] tp;
          tp = new T [nElem = e.nElem];
          for (int i = 0; i < nElem; i++)
                tp[i] = e.tp[i];
      }
      return *this;
}

#endif

#include <iostream>
using namespace std;

#include "feladat1a.hpp"

void main()
{
    try {
        String<char> s1;

        s1.insert(0, 'A');
        s1.insert(0, 'B');
        s1.insert(10, '0');
        s1.insert(0, 'C');

        String<char> s2 = s1;
        cout << s1[21];
    } catch (exception& e) {
        cerr << e.what();
    }
}
```

A javításnál nagyon negatívan értékeltük, ha nem a feladathoz tartozó kód került a megoldásokba. Ez ugyanis arra utal, hogy a legális puskáról lemásolt valamit, de fogalma sincs, hogy mit.

5. feladat egy lehetséges megoldása:



```
class Utasitas {
public:
    virtual bool Do() = 0;
};

class Mos :public Utasitas {
    bool Do();
};

class Oblit :public Utasitas {
    bool Do();
};

class Vezerlo {
     Utasitas *v[200];
     int  db;
public:
    Vezerlo() :db(0) {};
    int  getSzam() { return db; }
    void torol(int);          // utasítás törlése
    void beszur(Utasitas*, int); // utasítás beszúrása
    void start();
};

void Vezerlo::torol(int k) {
    for (int i = k; i < db-1; i++)
        v[i] = v[i+1];       // áthelyez
    db--;                    // csökkentjük a darabszámot (nem ell.)
}

void Vezerlo::beszur(Utasitas *u,  int k) {
    for (int i = db; i > k; i--)
        v[i] = v[i-1];       // áthelyez
    v[k] = u;                // beszúr (nem ellenőriz)
    db++;
}

void Vezerlo::start() {
    int i = 0;
    while (db && v[i]->Do()) {// addig amíg igazzal tér vissza
        i++;
        if (i >= db) i = 0; // körbefordul
    }
}
main() {
     Vezerlo v;
     v.beszur(new Mos, 0);
     v.beszur(new Oblit, 1);
     v.beszur(new Oblit, 2);
     v.start();
     return 0;
}
```