rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 3. beszámoló dolgozat. 2017.03.16. Kurz/**Terem**: Gy2/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt egy tetszőleges hosszúságú karaktersorozat tárolására készítettük:

```
class CharSeq {
    char *seq;
public:
    CharSeq() { seq = new char[1]; seq[0] = '\0';}
    CharSeq& operator+=(char ch) {
        int len = strlen(seq);
        char *tmp = new char[len+2];
        strcpy(tmp, seq); tmp[len] = ch; tmp[len+1] = '\0';
        delete[] seq;
        seq = tmp;
        return *this;
    }
    const char* c_str() const { return seq; }
    ~CharSeq() { delete[] seq; }
    CharSeq(const CharSeq& d) {
        seq = new char[strlen(d.seq)+1];
        strcpy(seq, d.seq);
    }
};
```

1. Magyarázza meg, hogy az alábbi kódrészlet futtatásakor miért lép fel memóriakezelési hiba! (1p)

```
{ CharSeq s; s += 'A'; s += 'B'; s += 'C';
  std::cout << s.c_str() << std::endl; }
```

Az alapértelmezett destruktor nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy a fenti kódrészlet futtatása ne okozzon hibát! Használja az üres helyeket az osztály deklarációjában! (1p)

Mit ír a szabványos kimenetre a fenti kódrészlet? ABC (1p)

2. Milyen problémára kell számítani, ha egy CharSeq osztályból létrehozott objektumot kivételként eldobunk? Mi a probléma forrása? (1p)

Kivétel dobásakor az osztály másoló konstruktora hívódik. Az alapértelmezett másoló konstruktor azonban nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy ne lépjen fel az a probléma, és az osztály az elvárásoknak megfelelően működjön! Használja az üres helyeket az osztály deklarációjában! (3p)

3. Készítsen egy inserter operátort, amivel egy **CharSeq** osztályból származó objektum kiírható egy std::ostream típusú objektumra. Működjön az elvárásoknak megfelelően az alábbi kódrészlet! (3p)

```
{ CharSeq d1; d1+= 'U'; std::cerr << d1 << std::endl; }
```

```
std::ostream& operator<<(std::ostream& os, const CharSeq& s) {
    return os << s.c_str();
}
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 3. ellenőrző dolgozat. 2017.03.17. Kurz/**Terem**: Gy2/	Elért pontszám:
Név: **Neptun:**	

Az alábbi osztályt tetszőleges hosszúságú szövegek tárolására készítettük:

```
class Duma {
    char *p;
public:
    Duma(const char *s = "")    { p = new char[strlen(s)+1]; strcpy(p, s); }
    const char* c_str() const   { return p; }
    char& operator[](size_t i) { return p[i]; }
    ~Duma() { delete[] p; }
    Duma(const Duma& d) {
        p = new char[strlen(d.p)+1];
        strcpy(p, d.p);
    }
```

```
};
```

1. Magyarázza meg, hogy az alábbi kódrészlet futtatásakor miért lép fel memóriakezelési hiba! (1p)

```
{ Duma s("Hello"); std::cout << s.c_str() << std::endl; }
```

Az alapértelmezett destruktor nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy a fenti kódrészlet futtatása ne okozzon hibát! Használja az üres helyeket az osztály deklarációjában! (1p)

Mit ír a szabványos kimenetre a fenti kódrészlet? Hello (1p)

2. Milyen problémára kell számítani, ha készítünk, és használunk egy **Duma** visszatérési értékű függvényt (pl. Duma fv())? Mi a probléma forrása? (1p)

Az objektum értékű függvény a visszatéréskor meghívja az objektum másoló konstruktorát. Az alapértelmezett másoló konstruktor azonban nem jó, mivel az osztály dinamikus memóriaterületet foglal és tart nyilván.

Módosítsa az osztályt, hogy ne lépjen fel az a probléma, és az osztály az elvárásoknak megfelelően működjön! Használja az üres helyeket az osztály deklarációjában! (3p)

3. Készítsen egy inserter operátort, amivel egy **Duma** osztályból származó objektum kiírható egy std::ostream típusú objektumra. Működjön az elvárásoknak megfelelően az alábbi kódrészlet! (3p)

```
{ Duma s1("C++"); std::cerr << s1 << std::endl; }
```

```
std::ostream& operator<<(std::ostream& os, const Duma& s) {
    return os << s.c_str();
}
```