

Demo C2 / 2.5

Vähintään kahteen vastaukseen ”automaattinen” testaus perinteisellä tavalla. Vastaukset [NettiDemoWWW](#):hen ”nimelle” c2.

1. Suunnittele luokka `cLinjaAuto`, jossa on paikkojen lukumäärä ja vapaiden paikkojen lukumäärä. Tee metodit `tulosta` sekä `lisaa` ja vahenna muuttamaan matkustajien lukumäärää. Alla testipääohjelma:

```
...
int main(void)
{
    cLinjaAuto pikkubussi(10),isobussi(45);
    pikkubussi.lisaa(4); pikkubussi.tulosta();
    isobussi.lisaa(30); isobussi.tulosta();
    int yli = pikkubussi.lisaa(15);
    isobussi.lisaa(yli);
    pikkubussi.tulosta(); isobussi.tulosta();
    if ( isobussi.tilaa() )
        cout << "Isoon bussiin mahtuu!" << endl;
    return 0;
}
```

2. Kirjoita luokka `cKulkuneuvo`, jossa on ainakin nopeus ja matkustajien lukumäärä. Peritstä luokat `cLaiva` ja `cLentokone` joissa kummassakin on jokin oma erikoisominaisuus yleiseen kulkuneuvoon verrattuna. Kirjoita myös pieni testipääohjelma.
3. Muuta [valuutts.cpp](#):n luokka `cValuutat` sellaiseksi, että perinnän tilalla käytetään koostamista. Mihinkään muualle ei saa tehdä muutoksia kuin tähän luokkaan. [Vinkkejä kääntämiseen](#).
- 4*. Kirjoita [taul_d.cpp](#) (ks. [moniste 16.5](#)) taulukolle metodi `sijoita`, jolla voidaan sijoittaa taulukkoon toinen taulukko:

```
...
int main(void)
{
    cTaulukko luvut(7);
    luvut.lisaa(0); luvut.lisaa(2);
    cout << luvut << endl; // 0 2
    cTaulukko taul;
    taul.sijoita(luvut); // tai jopa taul = luvut;
    cout << taul << endl; // tulostaa saman kuin edellä
    return 0;
}
```

Luokka voidaan testata vaikka seuraavasti [ComTestC++:lla](#)

```
#include <sstream>
using std::ostringstream;
...
```

```

/**
 * Luokka dynaamiselle int-taulukolle
 * @code
 * <pre name="test">
 *   ostream ss;
 *   cTaulukko luvut(10);
 *   ss << luvut; ss.str() === "";
 *   luvut.lisaa(0); luvut.lisaa(2); luvut.lisaa(2);
 *   luvut.lisaa(4); luvut.lisaa(2);
 *   ss << luvut; ss.str() === "0 2 2 4 2 ";          ss.str("");
 *   cTaulukko taul;
 *   taul.sijoita(luvut);
 *   ss << taul;  ss.str() === "0 2 2 4 2 ";          ss.str("");
 *   taul.lisaa(99);
 *   luvut.lisaa(88);
 *   ss << taul;  ss.str() === "0 2 2 4 2 99 ";      ss.str("");
 *   ss << luvut; ss.str() === "0 2 2 4 2 88 ";      ss.str("");
 * </pre>
 * @endcode
 */

```

- 5*. Kirjoita edellisen tehtävän luokkaan + operaattori, niin että seuraava koodi toimii:

```

cTaulukko t3 = luvut + taul;
cout << t3 << endl;      // 0 2 0 2

```

Arvostelee ratkaisusi tehokkuutta syntyvien olioiden näkökulmasta.

- 6-7. Kirjoita ohjelma, joka kysyy nimiä kunnes annetaan tyhjä rivi ja tulostaa tämän jälkeen nimet aakkosjärjestyksessä. Nimet syötetään muodossa "*Aku Ankka*", mutta siitä huolimatta ne lajitellaan siten, että "*Iines Ankka*" < "*Hannu Hanhi*", eli sukunimen mukaan. Käytä ratkaisussa jotakin STL:n tietorakennetta sekä algoritmia ja ota pohjaksi esim: [AakkostaHenkilot.java](#).

```

// Merkkijonon lukeminen pääteltä:
string s;
cout << "Anna jono";
getline(cin, s);

```

8. Muuta pasianssiohjelma ([pasi.cpp](#)) kunnolliseksi olio-ohjelmaksi, eli tietueet luokiksi ja aliohjelmat metodeiksi.
9. Muuta pasianssiohjelmaa siten, että ohjelma piirtää lopuksi histogrammin siitä, millä todennäköisyydellä peli päättyy *i*:teen korttiin (siis esimerkiksi seuraavasti):

```

1 *****
2 *****
3 *****
4 ****
5 **
6 *
7 *
...

```

Malliohjelmassa luokka näytti seuraavalta:

```
/*
class cHisto {
    int lkm;
    long *esiintymat;
public:
    cHisto(int n);
    ~cHisto();
    int lisaa(int i);
    long etsi_max() const;
    int piirra() const;
};
*/
```

- 10*. Lisää cHisto-luokkaan private sijoitusoperaattori ja CopyKonstruktori. (Tämän voi tehdä vaikka ei tekisi 8-9 tehtäviä, kun tekee "dummy"-metodit noista muista).