OHJ-1150 Ohjelmointi II – tentti 17.12.2009

tentin laatinut: Ari Suntioinen <*ari.suntioinen@tut.fi*>

Tentissä saa olla esillä kurssien OHJ-1100 Ohjelmointi I- ja OHJ-1150 Ohjelmointi II -luentomonisteet ja kurssin kotisivulta ladattava c++-kirjastoreferenssi, joissa saa olla käsin tehtyjä lisämerkintöjä. Muuta lisämateriaalia tai laskimia ei saa olla esillä.

Tehtävä 1

Vastaa ***ensimmäisenä*** tämän tehtävän kysymyksiin:

(a) Kirjoita nimesi ja opiskelijanumerosi ***selkeästi*** jokaisen palauttamasi paperin etusivun oikeaan yläkulmaan. [1 p]

(b) Kopioi seuraava taulukko siististi päällimmäiselle vastauspaperille nimesi ja opiskelijanumerosi alle siten, että jokainen "ruutu" on kooltaan 2×2 konseptipaperin ruutua:

1	2	3	4	5	Σ

[1 p]

Tehtävä 2

Selitä lyhyesti (max. 3–4 virkettä) seuraavat:

(a) rekursiivinen tietorakenne, [5 p]

(b) FIFO (first in first out), [5 p]

(c) kopiorakentaja, [5 p]

(d) miksi luokka ja moduuli tarkoittavat eri asiaa ja [5 p]

(e) julkisen rajapinnan ja abstratiorajan välinen yhteys. [5 p]

Huomioi vastatessasi seuraavat:

– Esimerkki ei yksinään riitä vastaukseksi: anna yleinen selitys.

– Älä selitä kysyttyä termiä sen itsensä (tai sen taivutusmuotojen) avulla.

– Selitä yksikäsitteisesti: jos vastauksesi voi tulkita väärin, se tulkitaan väärin.

– Älä kopioi tekstiä sana–sanalta monisteesta: se ei osoita asian ymmärrystä (tähän tehtävään kannattaa yrittää vastata ensin kurkkimatta monistetta).

Huomaa myös, että kaikkiin kohtiin ei välttämättä löydy sanatarkkaa selitystä monisteesta, vaan joudut kertomaan, miten termin merkitys on sinulle avautunut.

Tehtävä 3

Seuraavassa on määritelty rekursiivinen funktio `func`, joka on kommentoitu niin huonosti, että kommenteista ei voi päätellä sen toimintaa.

```
int func(vector<int>& vek, vector<int>::size_type i = 0) {
    if ( i == vek.size() - 1 ) {
        return vek.back();
    } else if ( vek.at(i) > vek.at(i + 1) ) {
        swap(vek.at(i), vek.at(i + 1)); // algorithm-kirjastosta
        return func(vek, i + 1);
    } else {
        return func(vek, i + 1);
    }
}
```

Tutki funktiota ja vastaa seuraaviin kysymyksiin:

(a) Mitä funktio palauttaa, jos sitä kutsutaan antamalla parametrina vektori, jossa on vain yksi alkio? [2 p]

(b) Entä jos sitä kutsutaan antamalla parametrina vektori, joka on tyhjä? [3 p]

(c) Mitä funktio palauttaa, jos parametrivektorin alkiot ovat kasvavassa järjestyksessä? [5 p]

(d) Mitä funktio yleisesti ilmaistuna tekee, jos parametrivektori on kooltaan jotain muuta kuin nolla tai yksi? [10 p]

(e) Onko funktio `func` häntärekursiivinen? [1 p]

(f) Perustelusi edellisen kohdan vastaukselle. [4 p]

Käännä ➠

Tehtävä 4

Ajatellaan seuraavanlaisista Solu-structeista

```
struct Solu {
    int   avain;
    int   data:
    Solu *seur;
    Solu *edel;
};
```

koostuvaa dynaamista linkitettyä rakennetta, jonka sijainnista muistissa pidetään kirjaa osoittimilla `eka` ja `vika`.
Rakenteelle on määritelty lisäysalgoritmi, jolla muuttujiin `uusi_avain` ja `uusi_data` talletetut arvot saadaan lisättyä rakenteeseen:

```
Solu *uusi_solu = new Solu;
uusi_solu->avain = uusi_avain;
uusi_solu->data  = uusi_data;
uusi_solu->seur = uusi_solu->edel  = 0;
if ( eka == 0 ) {
    eka = vika = uusi_solu;
} else {
    Solu *apu = eka;
    while ( true ) {
        if ( apu->avain == uusi_avain) {
            uusi_solu->edel = apu;
        }
        if ( apu->seur == 0 ) {
            break;
        }
        apu = apu->seur;
    }
    vika = apu->seur = uusi_solu;
}
```

Nämä määrittelyt huomioiden:

(a) Piirrä laatikko–nuoli-kaaviona tilanne, kun tyhjään rakenteeseen on lisätty seuraavassa järjestyksessä uudet (*avain*, *data*)-parit: (1, 1), (3, 1), (2, 1), (1, 2), (2, 2), (3, 2), (4, 1) ja (3, 3). [7 p]

(b) Mihin listan solun `edel`-kenttä osoittaa? [5 p]

(c) Esitä c++-koodi, joka tulostaa lisäysjärjestykselle vastakkaisessa järjestyksessä solut, joilla on sama avain-kenttä kuin viimeisellä solulla? [9 p]

(d) Esitä c++-koodi, joka tulostaa samat alkiot kuin edellisen kohdan koodi, mutta siinä järjestyksessä, kun ne on lisätty rakenteeseen. [4 p]

Täysien pisteiden saannin ehtona kahdessa viimeisessä kohdassa on se, ettei koko listaa käydä alkio alkiolta läpi.

Tehtävä 5

Pohditaan `Makefile`ä. Aihepiiri on kaikille tuttu kurssin tässä vaiheessa. Keskitytään tarkastelussa puhtaasti riippuvuussuhteiden esittämiseen ja unohdetaan päivityskomennot, makromäärittelyt yms.

`Makefile` kuvailee tiedostojen välisiä riippuvuuksia tekstuaalisessa muodossa. Esimerkiksi rivit:

```
tiedostoA: tiedostoB tiedostoC
tiedostoC: tiedostoB tiedostoD
```

kertovat, että `tiedostoA` riippuu tiedostoista `tiedostoB` ja `tiedostoC`, ja `tiedostoC` riippuu tiedostoista `tiedostoB` ja `tiedostoD`. Käytännössä tämä tarkoittaa sitä, että jos jompaa kumpaa tai molempia tiedostoista `tiedostoB` tai `tiedostoC` muutetaan, `tiedostoA` ei enää sen jälkeen olen ajan tasalla. Toisaalta myös muutos `tiedostoD`:ssä johtaa `tiedostoA`:n vanhenemiseen, koska se riippuu `tiedostoD`:stä epäsuoraan `tiedostoC`:n kautta. Lyhyesti sanottuna: `Makefile` kuvaa tiedostojen riippuvuussuhteiden verkon.

Suunnittele tietotyyppi, jota voidaan käyttää edellä esitettyjen riippuvuussuhdeverkkojen käsittelyyn c++-ohjelmassa.

Ainakin seuraavat asiat pitää huomioida:

(a) Tyypin julkisen rajapinnan määrittely, siis luokan public-osa (lue koko tehtävä ennen kuin vastaat tähän kohtaan). [4 p]

(b) Missä muodossa riippuvuudet tallennetaan private-osaan (a- ja b-kohdat voi yhdistää yhdeksi luokkamäärittelyksi). [6 p]

(c) c++-toteutus funktiolle, jonka avulla rakenteeseen voidaan lisätä yhden tiedoston riippuvuustiedot. [4 p]

(d) c++-toteutus funktiolle, jonka ***paluuarvona*** saadaan kaikki tiedostot, joista jokin kurssi riippuu suoraan. Aiempaa esimerkkiä soveltaen: jos tutkittava tiedosto on `tiedostoA`, tämä funktio palauttaisi `tiedostoB` ja `tiedostoC`, eli ne tiedostot, jotka `Makefile`:ssä on suoraan nimetty riippuvuusuhteiksi. [6 p]

(e) Toteuta c++-funktio, jonka paluuarvo on `true`, vain jos sen ensimmäisenä parametrina annettu tiedosto riippuu suoraan tai epäsuoraan toisena parametrina annetusta tiedostosta. Esimerkiksi paluuarvo olisi `true`, jos funtiolle annettaisiin parametreina `tiedostoA` ja `tiedostoD`. Tai toisaalta, paluuarvo olisi `false`, jos parametrit olisivat `tiedostoB` ja `tiedostoD`. [5 p]

Mitä selkeämpi ja elegantimpi ratkaisu, sitä paremmat pisteet. Voit käyttää kaikkia tuntemiasi työkaluja (tämän on takoitus olla STL-säiliöiden avulla toteutettava tehtävä).