Vastaa neljään kysymykseen! Tentin arvosteluasteikko on 0 – 24 pistettä. Kaikkien kysymysten painoarvo on sama (6 pistettä/tehtävä). **Lähtökohtana on, että kysymyksessä 2 käytetään C++-kieltä ja muissa C:tä.** Jos kuitenkin kysymyksessä 2 noppa on toteutettu oikeaoppisesti abstraktina tietotyyppinä C:llä, voi ko. tehtävästä saada 4 pistettä.

1.

Eräässä tietokoneympäristössä tietotyypeillä on seuraavat koot:

tyyppi	koko tavuina
char	1
int	2
double	8
osoitin	4

Seuraavassa on kirjoitettu 6 erilaista määrittelyä. **Selvitä kustakin määrittelystä seuraavat asiat**. 1) Kuinka paljon muistista varataan tilaa kyseisellä määrittelyllä. 2) Millä tavalla, mihin tarkoitukseen tai missä yhteydessä määriteltyä muuttujaa voisi käyttää.

a) `int *a[10];`
b) `double (*b)[10];`
c) `char **c;`
d) `int (*d[10] )[20];`
e) `double *(*e)(int);`
f) `double (**f)(int);`

2.

C:ssä voidaan generoida satunnaislukuja funktiolla rand. Kutsumalla tätä funktiota aina uudelleen saadaan uusi satunnaisluku satunnaislukujen sarjasta. Ennen kyseisen funktion kutsua pitää satunnaislukugeneraattorille antaa siemenluku funktiolla srand (tämä tehdään ohjelmassa kerran). Usein siemenluvuksi annetaan koneen ajanlaskennassa käytettävä funktion time palauttama arvo, jolloin srand funktiota kutsutaan muodossa srand(time(NULL));

Funktio rand voi antaa samoja lukuja uudelleen (samoin kuin 6-sivuinen noppa heitettäessä). Tämän takia se ei sellaisenaan sovi kaikkiin tilanteisiin. Siksi halutaan tehdä helppokäyttöinen satunnaislukugeneraattori luokkana Tnoppa. Tnoppa luokan nopalla voidaan generoida satunnaisia kokonaislukuja halutulta alueelta. Sille voidaan noppaa konstruoitaessa lisäksi kertoa, saako sama luku esiintyä sarjassa uudelleen vai ei. Lisäksi se voidaan välillä resetoida alkutilaan, jonka jälkeen siihen mennessä jo arvotut numerot eivät enää vaikuta resetoinnin jälkeisiin generaattorilta saataviin lukuihin. Resetoinnissa generaattorille annetaan myös uusi siemenluku, mutta lukualue, jolta luvut arvotaan säilyy (tällaista resetointia tarvitaan esim. tilanteessa, kun lähdetään arpomaan itselle uutta seuraavaa veikattavaa

Luokalla Tnoppa on siis jäsenfunktiot

Tnoppa (konstruktori)
heita
reset

Konstruktorilla asetetaan satunnaislukujen alueen alaraja ja yläraja, tarvittavan satunnaislukusarjan pituus (lukujen maksimimäärä), sekä tieto saako sarjassa esiintyä duplikaatteja vai ei (sama luku enemmän kuin kerran). Jäsenfunktiolla heita saadaan seuraava satunnaisluku sarjasta. Funktio reset resetoi nopan kuten yllä on esitetty.
Kirjoita luokan Tnoppa esittely ja sen kolmen jäsenfunktion toteutukset.

Huomautus 1. Funktion srand prototyyppi on void srand(unsigned int seed) ja funktion rand prototyyppi on int rand(void). Funktio palauttaa satunnaisen kokonaisluvun väliltä 0 – RAND_MAX (joka voi olla esim 32567).

3.

C-kielen standardikirjaston merkkijonojen käsittelyfunktioissa on yleensä lähtökohtana se, että tila mahdolliselle tulosmerkkijonolle on valmiiksi varattuna. Esimerkiksi funktion strcpy (merkkijonon kopiointifunktio), jonka prototyyppi on

char * strcpy(char *destination, const char * source);

tapauksessa parametrin destination osoittamalla muistialueella tulee olla riittävästi tilaa merkkijonon source kopiolle. C-kielen standardikirjastosta myös puuttuu joitakin käteviä mm. Basic-kielestä tuttuja funktioita kuten osan ottaminen merkkijonosta lähtien tietystä merkkipaikasta ottamalla tietty määrä merkkejä siitä eteenpäin.

Kirjoita C-kielen merkkijonoille seuraavat kolme funktiota:

substring
allocstrcpy
allocstrcat

Funktiolle substring välitetään kolme parametriä: 1) merkkijono, josta otetaan osa, 2) sen merkkipaikan indeksi (numero), josta lähtien alastringi otetaan ja 3) alastringin pituus eli otettavien merkkien määrä. Funktio palauttaa osoittimen alastringiin. Funktio varaa alastringille juuri sopivan kokoisen tilan dynaamisesta muistista. Alkuperäinen stringi säilyy siis muuttumattomana.

Funktiolla allocstrcpy voidaan tehdä kopio merkkijonosta. Ero standardikirjastossa olevaan funktioon strcpy on siinä, että allocstrcpy varaa

ensimmäisen parametrin edustaman merkkijonon perään. Tässäkin tapauksessa lähtökohta on se, että ei voida olettaa, että sen merkkijonon perässä, johon liitetään olisi tilaa ylimääräisille merkeille. Siksi funktio allocstrcat varaa juuri sopivan pituisen muistialueen tulosstringille. Samoin kuin standardifunktion strcat tapauksessa, merkkijono, jonka perään liitetään ei enää ole sellaisenaan erillisenä olemassa funktion käytön jälkeen.

Näyttääksesi kuinka kirjoittamiasi funktioita käytetään, **laadi pieni pääohjelma**, joka ensin lukee näppäimistöltä kaksi merkkijonoa (string1 ja string2). Sitten ohjelma tekee kopion ensimmäisestä luetusta merkkijonosta (copyString) dynaamiselle alueelle funktiolla allocstrcpy. Seuraavaksi ohjelma muodostaa uuden merkkijonon (leftString) 4:stä ensimmäisestä merkkijonon string2 merkistä funktiolla substring. Lopuksi ohjelma muodostaa tulosmerkkijonon (resultString) liittämällä ensimmäisen luetun merkkijonon kopioon copyString toisen luetun merkkijonon alun leftString. Tällöin siis molemmat alunperin luetut merkkijonot pysyvät muistissa muuttumattomina.

Huomautus. Standardifunktion strcat prototyyppi on

```
char * strcat(char *destination, const char * source);
```

4.

On olemassa abstrakti tietotyyppi piste, joka muodostuu tietotyypistä Tpoint ja sen operaatiofunktioista:

```
void setPoint(Tpoint *p, float x0, float y0);
float pointDistance(Tpoint p1, Tpoint p2);
void movePoint(Tpoint *p, float deltaX, float deltaY);
```

Tietotyyppi Tpoint on määritelty tietueeksi, joka sisältää pisteen paikan tasolla määrittämiseen tarvittavat tiedot (tässä x- ja y-koordinaatti). Funktio setPoint asettaa pisteen koordinaattiarvoiksi parametrina annetut arvot (x0, y0). Funktio pointDistance laskee kahden pisteen välisen etäisyyden ja funktio movePoint siirtää pisteen paikkaa parametrien deltaX ja deltaY ilmoittamien suhteellisten siirtymien verran.

Ympyrän tietoja ovat säde ja keskipiste. Ympyrän tietotyyppiin ei tällä kertaa sisällytetä keskipistettä kuvaavaa pistettä itseään, vaan vain osoitin pisteeseen. Tila keskipisteelle varataan dynaamisesta muistista ympyrän muodostamisen yhteydessä. Ympyrän muodostamiseen tehdään operaatiofunktio constructCircle. Ympyrällä on kaikkiaan seuraavat operaatiofunktiot:

1) constructCircle, jolle voidaan antaa parametrina ympyrän keskipisteen x-koordinaatti, y-koordinaatti ja sade. Funktio varaa tilan pisteelle, asettaa sille alkuarvot ja "kytkee" pisteen ympyrän keskipisteeksi.
2) moveCircle, joka siirtää ympyrän paikkaa (parametrina Δx ja Δy).

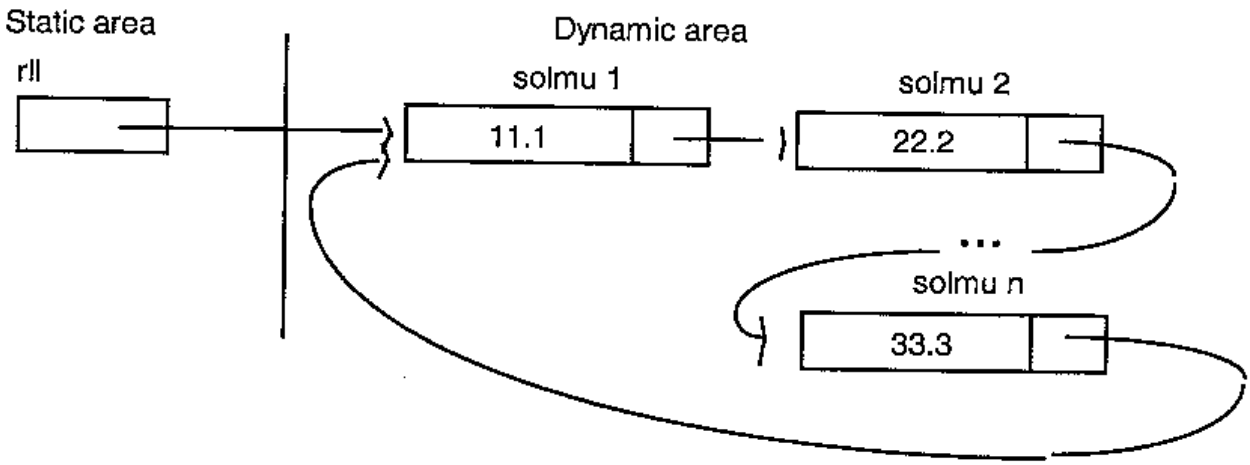
huomioiden, jolloin etäisyys voi siis olla negatiivinen).
4) destructCircle, joka vapauttaa keskipisteelle varatun tilan.

Toteuta ADT ympyrä ylläesitettyjen periaatteiden mukaisesti eli määrittele tietotyyppi TCircle ja toteuta sen neljä operaatiofunktiota.

Huomautus. Ympyrän operaatioita toteutettaessa on käytettävä hyväksi annettua adt pistettä. Pisteelle ei kirjoiteta toteutusta.

5.

Kirjoita tietomäärittelyt, jotka tarvitaan alla kuvatun renkaaksi linkitetyn listan (rll) kuvaamiseen. Renkaaksi linkitetyn listan solmuissa (node) tietokenttä (data field, item field) on floating point luku. Huomaa, että yksi osoitin edustaa rengasta. **Kirjoita funktio laske_solmut**, joka laskee ja palauttaa tiedon, kuinka monta solmua parametrina annettu renkaaksi linkitetty lista sisältää. **Kirjoita lisäksi funktio lisaa_solmu**, joka lisää parametrina annettuun renkaaseen yhden solmun siten, että renkaan solmuosoitin osoittaa lisäyksen jälkeen uuteen solmuun. Uuden solmun tietosisältö annetaan funktiolle parametrina.



Huom 1. Erikoistapaukset pitää tietysti hoitaa asianmukaisesti (tyhjä lista, vain yksi alkio listassa jne). NULL-osoitin edustaa tyhjää listaa.

Huom: Saat yhden lisäpisteen antamalla palautetta kurssista kurssin kotisivujen (kurssinhallintajärjestelmän) kautta.

https://as0101.hut.fi/students/feedback_form.php