AS-0.1101 C-ohjelmoinnin peruskurssi / Tentti 29.08.2006 / Aki Hiisilä

Vastaa viiteen kysymykseen! Tentin arvosteluasteikko on 0 – 30 pistettä. Kaikkien kysymysten painoarvo on sama (6 pistettä/tehtävä). Ohjelmointitehtät tulee kirjoittaa C-kielellä hyvää ohjelmlointityyliä noudattaen.

Palauta kolme konseptia, siten että ensimmäisellä konseptilla on vastaukset tehtäviin 1 ja 2, toisella konseptilla vastaukset tehtäviin 3 ja 4, sekä kolmannella konseptilla vastaukset tehtäviin 5 ja 6.

Tehtävä 1

Mitä alla oleva ohjelma tulostaa?

```
#include <stdlib.h>
#include <stdio.h>

int main(void)
{
  int **matrix;

  matrix = malloc(5 * sizeof(int *));
  *matrix = malloc(20 * sizeof(int));

  for(int i=0; i<5; i++){
    matrix[i] = matrix[0] + 4 * i;

    for(int j=0; j<4; j++)
      matrix[i][j] = 10 * i + j;
  }

  printf("a: %d\n", matrix[1][1]);
  printf("b: %d\n", *(matrix[3]));
  printf("c: %d\n", matrix[0][11]);
  printf("d: %d\n", matrix[3] - matrix[1]);
  printf("e: %d\n", *(*(matrix + 1) + 2));
  matrix[4] = matrix[3];
  printf("f: %d, %d\n", matrix[4][3],  matrix[0][19]);

  free(*matrix);
  free(matrix);
  return EXIT_SUCCESS;
}
```

Tehtävä 2

Tietty määrä opiskelijoita täyttää kysymyslomakkeen, jossa on 25 kysymystä. Jokaiseen kysymykseen vastataan antamalla vaihtoehdoksi luku 1, 2, 3, 4 tai 5. **Laadi ohjelma**, joka ensin kysyy vastanneiden opiskelijoiden lukumäärän. Tämän jälkeen se lukee annetut vastaukset siten, että ensin luetaan ensimmäisen opiskelijan vastaukset lähtien kysymyksestä 1 ja päätyen kysymykseen 25, sitten toisen opiskelijan antamat vastaukset samalla tavalla jne. Kun kaikkien opiskelijoiden vastaukset on syötetty, ohjelma selvittää mihin kysymyksiin kaikki ovat vastanneet samalla tavalla. Lopuksi ohjelma tulostaa niiden kysymysten numerot, joihin kaikki ovat vastanneet samalla tavalla.

Huomautus 1: Tehtävä pitää funktioilla jakaa tehtäväkuvauksen mukaisiin tai muuten järkeviin osiin ja tietojen välitys on tehtävä parametreilla. Globaaleja muuttujia ei saa käyttää.

Huomautus 2: Syöttötiedoille ei tarvitse tehdä muuta järkevyystarkastusta kuin, että vastausnumero on välillä 1…5.

Huomautus 3: Ohjelman käyttäjälle ei tarvitse antaa mitään ohjeita tai kehotteita.

Tehtävä 3

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

```
void pointInitialize(TPoint **p, float x0, float y0);
float pointDistance(const TPoint p1, const TPoint p2);
void pointMove(TPoint *p, float deltaX, float deltaY);
void pointDestruct(TPoint *p);
```

Tietotyyppi TPoint on määritelty tietueeksi, joka sisältää pisteen paikan tasolla määrittämiseen tarvittavat tiedot (tässä x- ja y-koordinaatti). Funktio pointInitialize varaa muistia pisteelle ja asettaa pisteen koordinaattiarvoiksi parametrina annetut arvot (x0, y0). Funktio pointDistance laskee kahden pisteen välisen etäisyyden ja funktio pointMove siirtää pisteen paikkaa parametrien deltaX ja deltaY ilmoittamien suhteellisten siirtymien verran. pointDestruct vapauttaa pisteelle varatun tilan.

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 circleInitialize. Ympyrällä on kaikkiaan seuraavat operaatiofunktiot:

circleInitialize, jolle voidaan antaa parametrina ympyrän keskipisteen x-koordinaatti, y-koordinaatti ja sade. Funktio varaa tilan ympyrälle ja asettaa sille keskipisteen ja säteen.

circleMove, joka siirtää ympyrän paikkaa (parametrina deltaX ja deltaY).

circleDistance, joka laskee kahden ympyrän välisen etäisyyden (säteet huomioiden, jolloin etäisyys voi siis olla negatiivinen).

circleDestruct, joka vapauttaa ympyrälle 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.

Tehtävä 4

Toteuta seuraavat merkkijonoja käsittelevät funktiot:

```
/** Kääntää merkkijonon toisinpäin (esim. ABC -> CBA).
 *  @param str Merkkijono joka käännetään.
 */
void reverse(char *str);

/** Lisää (liittää) merkkijonon n merkkijonon str perään.
 *  @param str Osoitin merkkijonoon.
 *  @param n Merkkijono, joka lisätään merkkijonon str perään.
 *  @return 0 jos onnistui, 1 virhetilanteissa.
 */
int addToEnd(char **str, const char *n);

/** Poistaa kaikki merkin c ilmentymät merkkijonosta str.
 *  @param str Osoitin merkkijonoon.
 *  @param c Merkki, jonka ilmentymät poistetaan.
 *  @return Kuinka monta merkkiä poistettiin tai -1 virhetilanteissa.
 */
int removeChars(char **str, char c);
```

Huomautus 1: Funktioiden pitää pitää huoli siitä, että merkkijonolle on varattuna aina täsmälleen oikea määrä muistia.

Tehtävä 5

Mitä alla oleva ohjelma tulostaa, kun oletetaan, että yksi tavu on kahdeksan bittiä?

```
#include <stdlib.h>
#include <stdio.h>

void print_1(unsigned char byte);
void print_2(unsigned char byte);

int main(void)
{
  unsigned char byte = 0x73;

  printf("A: ");    print_1(~byte);
  printf("\nB: "); print_1(byte & 0x8A);
  printf("\nC: "); print_1(byte || 0xFA);
  printf("\nD: "); print_1(byte >> 2);
  printf("\nE: "); print_2(byte);
  printf("\nF: "); print_2(byte ^ 0x71);
  printf("\n");
  return EXIT_SUCCESS;
}

void print_1(unsigned char byte)
{
  unsigned char mask;

  for(mask = 0x80; mask > 0; mask = mask >> 1) {

    if(byte & mask)
      printf("X");
    else
      printf("o");
  }
}

void print_2(unsigned char byte)
{
  if(byte) {
    print_2(byte >> 1);
    printf("%c", (byte & 0x01) ? 'X' : 'o');
  }
}
```

Tehtävä 6

Sovelluksessa tarvitaan paljon funktioita, joilla kaikilla on prototyyppi muotoa

```
double func(double a, double b);
```

Funktiot (niiden osoitteet) tallennetaan taulukkoon, josta niitä on helppo hakea indeksillä (eli niiden paikkanumerolla taulukossa). Funktiotaulukon luonti, yksittäisen funktion tallentaminen taulukkoon ja funktion haku taulukosta halutaan tehdä selkeillä funktioilla, joiden nimet ovat *funcArrayCreate*, *funcArraySetFunc* ja *funcArrayGetFunc*.

Funktiolle *funcArrayCreate* annetaan parametriksi funktioiden maksimimäärä. Se varaa tilan funktiotaulukolle ja palauttaa funktiotaulukon osoitteen.

Funktiolle *funcArraySetFunc* annetaan parametreina funktiolta *funcArrayCreate* saatu funktiotaulukko, funktio joka tallennetaan funktiotaulukkoon ja paikan indeksi taulukossa, joka ilmoittaa mihin kohtaan taulukossa funktio tallennetaan. Funktio *funcArraySetFunc* yksinkertaisesti vain tallentaa funktion funktiotaulukkoon.

Funktiolle *funcArrayGetFunc* annetaan parametreina funktiotaulukko ja paikan indeksi taulukossa, josta funktio haetaan. Funktio *funcArrayGetFunc* palauttaa funktion funktiotaulukosta parametrin ilmoittamasta paikasta.

Kirjoita yllämainitut kolme funktiota ja lyhyt pääohjelma, jolla näytät kuinka funktioita käytetään. Pääohjelmassa luodaan kahden funktion taulukko siten, että alkioon 0 tallennetaan funktio `double sum(double a, double b)` ja alkioon 1 funktio `double mul(double a, double b)`. Sitten pääohjelmassa haetaan funktio taulukon alkiosta 1 ja kutsutaan sitä parametreilla 6 ja 7, jolloin siis tulee suoritetuksi lukujen 6 ja 7 kertolasku, koska oletetaan, että funktio *mul* suorittaa kertolaskun.

Huomautus: Funktioita *sum* ja *mul* ei tarvitse kirjoittaa, koska ne ovat itsestään selviä.