C-ohjelmoinnin peruskurssi, Tentti 28.5.2015

Lyhyt referenssi funktioista tehtäväpaperin lopussa. Paperilla on 5 tehtävää, joista useimmissa on muutama alikohta. Maksimipistemäärä on 30 pistettä.

Kirjoita vastaukset konseptipapereille seuraavasti: tehtävät 1 ja 2 yhdelle konseptiarkille, tehtävät 3 ja 4 toiselle konseptille, ja tehtävä 5 omalle konseptilleen. Merkitse konseptille tehtävän numero selkeästi, ja kirjoita selkeällä käsialalla. Muista kirjoittaa oma nimi ja opiskelijanumero jokaiselle konseptille.

1. Mitä seuraava ohjelma tulostaa? Vastaukseksi riittää yksi rivi joka esittää tulosteen. (6 p)

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

int main(void)
{
    int a; for (a = 0; a < 5; a++);
    int b = (a > 5);
    char c[20];
    strcpy(c, "Hui");
    strcat(c, "Hai");
    int taul[3] = {10, 100, 200};
    int *d = taul + 1;
    int e = d[1];
    int f = 0xaa00 | 0x00bb;

    printf("a: %d  b: %d  c: %s  d: %d  e: %d  f: %x",
          a, b, c, *d, e, f);
    return 0;
}
```

2. Toteuta vastauspaperille seuraavat funktiot.

a) **`unsigned int pickmax(void)`**, joka lukee käyttäjältä etumerkittömiä kokonaislukuja kunnes käyttäjä antaa luvun 0, tai syötteen joka ei ole kokonaisluku. Funktio palauttaa annetuista kokonaisluvuista suurimman. (2p)

b) **`unsigned int strlen(const char *str)`**, joka palauttaa annetun merkkijonon pituuden. Et saa käyttää toteutuksessa string.h – otsakkeessa määriteltyä saman nimistä funktiota. (2 p)

c) **`void printbits(unsigned char num)`**, joka tulostaa annetun 8-bittisen luvun binääriesityksenä ’1’ ja ’0’ – merkkejä käyttäen, eniten merkitsevä bitti ensin. Esimerkiksi **`printbits(0xD5)`** tulostaa ”**`11010101`**”. (2p)

3. Seuraavissa funktioissa on virheitä. Kerro kustakin virheestä virheellisen rivin numero, virheen syy lyhyesti, ja anna korjattu versio kyseisestä ohjelmarivistä. Mikäli haluat lisätä uuden rivin, kerro minkä rivin jälkeen se tulee lisätä. (Kolme kohtaa, huomaa c-kohta seuraavalla sivulla).

a) funktio, joka hakee merkkijonosta **str** merkkiä **c** ja palauttaa osoittimen siihen. Mikäli merkkiä ei löydy, palautetaan NULL. Funktio etsii myös loppumerkin, eli jos **c**:ksi annetaan '\0' palautuu osoitin siihen. (2 p)

```
1: char *mystrchr(char *str, char c)
2: {
3:     do {
4:         if (str == c) return str;
5:     } while (str++);
6:     return NULL;
7: }
```

Käännettäessä tulee seuraavanlainen ilmoitus:

```
tentti.c:4:10: warning: comparison between pointer and integer
```

Kun funktiota suoritetaan merkillä, jota annetusta merkkijonosta ei löydy, valgrind valittaa seuraavaa:

```
==627== Invalid read of size 1
==627==    at 0x4004C5: mystrchr (tentti.c:4)
==627==    by 0x40050B: main (tentti.c:39)
==627==  Address 0x401000 is not stack'd, malloc'd or (recently) free'd
```

b) funktio joka varaa muistista dynaamisen taulukon **size** kokonaisluvulle ja alustaa sen numeroilla jotka kasvavat 0:sta ylöspäin. Funktio palauttaa osoittimen luodun taulukon alkuun. Älä murehdi muistin vapauttamisesta: kutsuvan funktion oletetaan olevan siitä vastuussa. Voit (tässä tehtävässä) olettaa, että muistin varaus onnistuu aina. (2 p)

```
10: #include <stdlib.h>
11:
12: int *createArray(unsigned int size)
13: {
14:   int *array = malloc(size);
15:   for (unsigned int i = 0; i < size; i++) {
16:         array[i] = i;
17:   }
18:   return *array;
19: }
```

Jatkuu seuraavalla sivulla...

Kääntäjä (gcc) palauttaa seuraavan varoituksen:

```
testi2.c: In function ‘createArray’:
testi2.c:18:2: warning: return makes pointer from integer without a cast
```

Valgrind palauttaa funktiota suorittaessa seuraavaa:

```
==25876== Invalid write of size 4
==25876==    at 0x400616: createArray (testi2.c:16)
==25876==    by 0x40069C: main (testi2.c:44)
==25876==  Address 0x51ba048 is 8 bytes inside a block of size 10 alloc'd
==25876==    at 0x4C28BED: malloc (vg_replace_malloc.c:263)
==25876==    by 0x4005F3: createArray (testi2.c:14)
```

c) funktio, joka lisää dynaamisesti varattun taulukon (**array**) loppuun yhden uuden (`struct products` – tyyppisen) alkion, ja alustaa lisätyn (eli viimeisen alkion) parametrien **newtitle** ja **newprice** kertomalla sisällöllä. Taulukon aiempi koko on kerrottu parametrilla **length**. Funktio palauttaa osoittimen kasvatetun taulukon alkuun. Voit olettaa että muistin varaus onnistuu aina. (2 p)

```
20: #include <stdlib.h>
21: #include <string.h>
22:
23: struct products {
24:     char *title;
25:     double price;
26: };
27:
28: struct products *add_product(struct products *array, unsigned int length,
29:                           const char *newtitle, double newprice)
30: {
31:     struct products *np = realloc(array,
32:                           (length+1) * sizeof(struct products));
33:     strcpy(np->title, newtitle);
34:     np->price = newprice;
35:     return np;
36: }
```

Valgrind palauttaa funktiota suorittaessa seuraavaa:

```
==26955== Use of uninitialised value of size 8
==26955==    at 0x4C2953C: strcpy (mc_replace_strmem.c:429)
==26955==    by 0x4009D2: add_product (tentti.c:33)
==26955==  Uninitialised value was created by a heap allocation
==26955==    at 0x4C28BED: malloc (vg_replace_malloc.c:263)
==26955==    by 0x4C28D6F: realloc (vg_replace_malloc.c:632)
==26955==    by 0x4009B8: add_product (tentti.c:31)
==26955==
==26955== Invalid write of size 1
==26955==    at 0x4C2953C: strcpy (mc_replace_strmem.c:429)
==26955==    by 0x4009D2: add_product (tentti.c:33)
==26955==  Address 0x0 is not stack'd, malloc'd or (recently) free'd
```

Ohjelma myös keskeytyy Segmentation fault – signaaliin rivillä 33.

4. Mitä seuraavat funktiot (function_A, function_B, function_C) tekevät? Älä kuvaile toiminnallisuutta rivi riviltä, vaan kuvaile lyhyesti (1-2 lausetta) mutta täsmällisesti funktion tarkoitus ja mahdollinen paluuarvo. Jos funktio tulostaa jotain, kerro mitä se tulostaa. (2 pistettä kunkin funktion oikeasta ja täsmällisestä kuvauksesta)

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

int **function_A(unsigned int a, unsigned int b)
{
    int **r = malloc(a * sizeof(int *));
    for (unsigned int a2 = 0; a2 < a; a2++) {
      r[a2] = malloc(b * sizeof(int));
      for (unsigned int b2 = 0; b2 < b; b2++) {
          r[a2][b2] = a2 * b2;
      }
    }
    return r;
}


void function_B(char *b, const char *a)
{
    while(*a) {
      if (!(*a & (1 << 7)))
          *b++ = *a;
      a++;
    }
    *b = 0;
}


const char *function_C(const char *s1, const char *s2)
{
    const char *p = s1;
    unsigned int a = strlen(s2);

    for (; (p = strchr(p, *s2)) != 0; p++) {
      if (strncmp (p, s2, a) == 0)
          return p;
    }
    return NULL;
}
```

5. Suunnittele ilmoittautumisjärjestelmä kuvitteellista kurssia varten. Ilmoittautumisjärjestelmä listaa kurssille osallistuvat opiskelijat ja heidän opiskelijanumeronsa. Sinun tulee käyttää ja laajentaa seuraavia tietorakenteita:

```
typedef struct {
    char *name;  // name of student
    char ID[7];  // student ID (nul terminated)
    // you may add members to this structure if needed
} Student;

typedef struct {
    // add members here
} Course;
```

Course – rakenne listaa joukon **Student** - tietorakenteita jollain tapaa, mutta saat itse päättää miten. Toteutus voi olla esimerkiksi linkitetty lista tai dynaamisesti varattava taulukko. Täydennä tietorakenteet valmiiksi ja kirjoita ne paperille. Valmiiksi annettuja kenttiä **Student** - rakenteessa ei saa muuttaa, vaan niitä tulee käyttää sellaisinaan.

Kirjoita lisäksi toteutukset seuraaville funktioille:

a) **`void add_student(Course *c, const char *name, const char *ID)`** – joka lisää opiskelijan nimeltä **name**, opiskelijanumeroltaan **ID** opiskelijalistaan, jonka Course – tietorakenne (**c**) esittää.

b) **`void remove_student(Course *c, const char *ID)`** – joka poistaa opiskelijalistasta (**c**) opiskelijan tunnuksella **ID.** Voit olettaa että kukin opiskelijanumero esiintyy listassa vain kerran.

Toteutukset eivät saa käyttää enempää muistia kuin sen hetkisen opiskelijalistan ylläpitoon tarvitaan. Toisaalta niiden tulee kyetä tallentamaan suuriakin opiskelijamääriä. Toteutuksen tulee siis luottaa dynaamiseen muistinhallintaan. Voit olettaa että muistin määrä ei ole ongelma, ja muistinvaraukset onnistuvat aina.

Toimivien tietorakenteiden määrittelystä saat kaksi pistettä, sekä kummankin funktion toimivasta toteutuksesta kaksi pistettä.

Mahdollisesti hyödyllisiä funktioita

Merkkijonojen käsittelyyn (määritelty string.h - otsakkeessa):

- `size_t strlen(const char *s);` palauttaa annetun merkkijonon **s** pituuden.
- `char *strcpy(char *dest, const char *src);` kopioi merkkijonon **src** paikkaan **dest**
- `char *strncpy(char *dest, const char *src, size_t n);` kopioi enintään **n** merkkiä merkkijonosta **src** merkkijonoon **dest**. Jos merkkijono on lyhyempi kuin **n**, loput tavut täytetään '\0' - merkillä.
- `char *strcat(char *dest, const char *src);` liittää merkkijonon **src** merkkijonon **dest** perään
- `char *strchr(const char *s, int c);` etsii merkkijonosta **s** ensimmäisen kohdan jossa **c** esiintyy ja palauttaa osoittimen siihen. NULL jos **c** ei löydy
- `int strncmp(const char *s1, const char *s2, size_t n);` palauttaa 0 jos annetut merkkijonot ovat samat ensimmäisten **n** merkin osalta

Muistinhallinta (määritelty stdlib.h - otsakkeessa, memset string.h:ssa):

- `void *malloc(size_t size);` Varaa **size** tavua muistia, palauttaa osoitteen varattuun muistialueeseen
- `void *calloc(size_t nmemb, size_t size);` Varaa **nmemb** kertaa **size** tavua muistia, nollaa varatun muistialueen
- `void *realloc(void *ptr, size_t size);` Muutaa muistialueen **ptr** koon **size**:ksi, palauttaa uuden osoittimen muistialueeseen
- `void free(void *ptr);` vapauttaa annetun muistialueen
- `void *memset(void *s, int c, size_t n);` asettaa muistialueen **s**, jonka koko on **n**, kaikki tavut **c**:ksi

Merkkien käsittelyyn (määritelty ctype.h - otsakkeessa):

- `int toupper(int c);` muuta merkki isoksi kirjaimeksi
- `int tolower(int c);` muuta merkki pieneksi kirjaimeksi
- `int isalnum(int c);` onko merkki kirjain tai numero?
- `int isalpha(int c);` onko merkki kirjain?
- `int isspace(int c);` onko merkki tyhjä väli?
- `int islower(int c);` onko merkki pieni kirjain?
- `int isupper(int c);` onko merkki iso kirjain?

Muotoiltu I/O (määritelty stdio.h - otsakkeessa):

- `int printf(const char *format, ...);` tulostaa muotoiltua tulostetta annetusta merkkijonosta ja parametreista
- `int scanf(const char *format, ...);` lukee muotoiltua syötettä annettuihin osoitteisiin. Parametrit ovat siis muistiosoitteita

Muotoilumääreitä printf- ja scanf-funktioihin:

- %d: kokonaisluku
- %f: liukuluku
- %u: etumerkitön kokonaisluku
- %x: heksadesimaaliluku
- %c: merkki
- %s: merkkijono

Lukuja binäärimuodossa

0x0: 0000	0x8: 1000
0x1: 0001	0x9: 1001
0x2: 0010	0xA: 1010
0x3: 0011	0xB: 1011
0x4: 0100	0xC: 1100
0x5: 0101	0xD: 1101
0x6: 0110	0xE: 1110
0x7: 0111	0xF: 1111