C-ohjelmoinnin peruskurssi, Tentti 25.8.2014

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 = 4 << 1;
        int b; for(b = 10; b > 5; b = b - 6);
        char c[20]; int i = 5; c[0] = 0;
        while (i--) strcat(c, "!");
        double d = 1.0 / 3;
        int e[5]; e[0] = 1;
        int *pe = e;
        do {
                *(pe + 1) = *pe + *pe; pe++;
        } while (pe != &e[5]);
        int f = 0xaabb & 0xf0f0;
        printf("a: %d   b: %d   c: %s   d: %3.1f  e: %d  f: %x",
                a, b, c, d, e[4], f);
}
```

2. Toteuta vastauspaperille seuraavat funktiot.

a) **`double calc_average(const double *arr)`**, joka saa parametrinaan osoittimen taulukkoon, joka loppuu arvoon 0. Funktio palauttaa taulukossa olevien arvojen keskiarvon (viimeistä 0-alkiota ei lasketa mukaan). (2p)

b) **`int read_binary(const char *bits)`**, joka saa parametrinaan 8:n merkin merkkijonon, joka koostuu merkeistä '1' ja '0', ja palauttaa merkkijonossa esitetyn binääriluvun paluuarvonaan. Esim. "00010001" palauttaa 17 (eli 0x11 heksadesimaalimuodossa). (2 p)

c) **`int **multi_table(int xs, int ys)`**, joka varaa muistista kaksiulotteisen taulukon ja täyttää taulukon kertotaululla siten että alkion [i][j] arvo on (i+1) * (j+1). Taulukon mitat ovat **xs * ys**, eli taulukko sisältää arvoja välillä 1*1 -- xs * ys. Funktio palauttaa osoittimen varattuun taulukkoon. (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 vaihtaa annetun merkkijonon (**str**) pienet kirjaimet isoiksi, ja isot kirjaimet pieneksi. (2 p)

```
1: #include <ctype.h>
2: void switch_case(char *str)
3: {
4:     if (isupper(*str))
5:         *str = tolower(*str);
6:     else if (islower(*str))
7:         *str = toupper(*str);
8: }
```

b) Funktio joka saa parametreinaan taulukon merkkijonoja (**argv**), jossa on **argc** merkkijonoa, ja palauttaa pisimmän merkkijonon pituuden. Funktiossa on ainakin kaksi virhettä tai puutetta. (2 p)

```
30: int longest(int argc, char *argv[])
31: {
32:     int lc;
33:     for (int i = 0; i < argc; i++) {
34:         int sl = sizeof(argv[i]);
35:         if (sl > lc) {
36:             lc = sl;
37:         }
38:     }
39:     return lc;
40: }
```

Valgrind palauttaa funktiota suorittaessa seuraavaa:

```
==10788== Conditional jump or move depends on uninitialised value(s)
==10788==    at 0x400C54: longest (tentti-2014-08-25.c:35)
==10788==    by 0x401034: main (tentti-2014-08-25.c:232)
```

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<stdio.h>

int function_A(const char *a)
{
	FILE *f = fopen(a, "r");
	if (!f) return -1;
	int c;
	int d = 0;
	while ((c = fgetc(f)) != EOF) {
		printf("%02x ", c);
		d++;
		if (d % 8 == 0)
			fputc('\n', stdout);
	}
	return d;
}


void function_B(unsigned char *b)
{
	while (*b) {
		if (*b & (1 << 7))
			*b = '?';
		b++;
	}
}


const char *function_C(const char *a, const char *b)
{
	const char *ob = b;
	const char *ra = a;
	while (*a) {
		if (*ob == 0)
			return ra;
		if (*ob == *a)
			ob++;
		else {
			ob = b;
			ra = a+1;
		}
		a++;
	}
	return NULL;
}
```

5. Toteuta yksinkertainen jonotusjärjestelmä ohjelmointikurssin harjoituksia varten, jota käsitellään kahdella funktiolla seuraavasti:

a) `void enque(struct queue *q, const char *name)`, joka lisää jonon 'q' loppuun opiskelijan nimeltä 'name'.

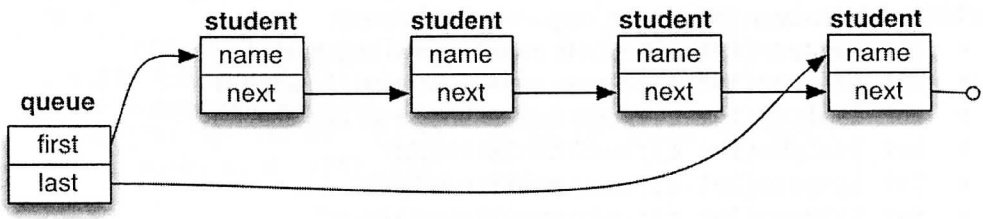
b) `struct student *deque(struct queue *q)`, joka poistaa jonon 'q' ensimmäisen alkion, ja palauttaa sen paluuarvonaan.

Edellä esitellyt funktiot käyttävät seuraavanlaisia tietorakenteita:

```
struct student {
	char name[20];
	struct student *next;
};

struct queue {
	struct student *first;
	struct student *last;
};
```

Jono on siis toteutettu linkitettynä listana, jossa kukin alkio on **'struct student'** -tyyppiä. **'struct queue'** ylläpitää osoitinta listan ensimmäiseen alkioon (**first**), ja listan viimeiseen alkioon (**last**). Lähtötilanteessa, kun lista on tyhjä, sekä first että last voidaan olettaa olevan NULL. Kun listassa on vain yksi alkio, molempien kenttien tulisi osoittaa samaan alkioon.



Funktio **'enque'** varaa tarvittavan muistin uudelle alkiolle (**struct student**) perustuen annettuun nimeen (**name**) ja linkittää sen **struct queue** - listan viimeiseksi alkioksi. Huomioi, että name-kentän pituus on rajattu. Kannattaa muistaa myös next-osoittimien päivittäminen.

Funktio **'deque'** palauttaa jonon ensimmäisen alkion ja poistaa sen linkitetystä listasta. Tämän johdosta **'first'**-osoitin muuttuu. deque-funktion ei tarvitse vapauttaa muistia, vaan oletetaan että funktion kutsuja vapauttaa muistin aikanaan.

Voit olettaa että muistin varaus onnistuu aina.

Tehtävänä on siis toteuttaa kaksi edellä mainittua funktiota.
Onnistunut enque-funktion toteuttaminen tuottaa maksimissaan 3 pistettä.
Onnistunut deque-funktion toteuttaminen tuottaa maksimissaan 3 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 *strncat(char *dest, const char *src, size_t n);` liittää enintään **n** merkkiä merkkijonosta **src** merkkijonoon **dest**
- `int strcmp(const char *s1, const char *s2);` palauttaa 0 jos annetut merkkijonot ovat samat, erisuuri kuin 0 jos merkkijonot eroavat

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