C-ohjelmoinnin peruskurssi, Tentti 16.5.2016

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 = (5 < 7);
	int b; for (b = 0; b == 10; b++);
	int c = strlen("text");
	int d = 1 << 1;
	int e = 0xACDC & 0x00FF;
	int arr[] = {10, 100, 200}; int *p = arr; p++; int f = *p;
	printf("a: %d, b: %d, c: %d, d: %d, e: %x, f: %d\n",
		a, b, c, d, e, f);
}
```

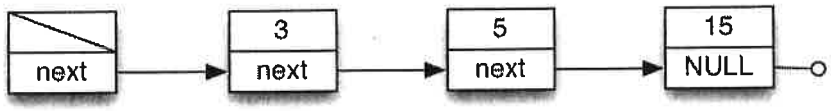
2. Toteuta vastauspaperille seuraavat funktiot.

a) **`char *mystrcat(char *dest, const char *src)`**, joka lisää merkkijonon *src* merkkijonon *dest* perään. Funktio palauttaa osoittimen yhdistetyn merkkijonon alkuun. (2p)

b) **`int *read_numbers(void)`**, joka lukee käyttäjältä positiivisia kokonaislukuja ja lisää kunkin luvun dynaamisesti varatun taulukon loppuun. Lukeminen lopetetaan kun käyttäjä syöttää 0:n tai negatiivisen luvun. Funktio palauttaa osoittimen dynaamisesti varattuun taulukkoon. (2p)

c) **`void set_bit(unsigned char *buffer, unsigned int n, int bit)`**, joka käy läpi *n* tavua alkaen osoitteesta *buffer*, ja asettaa bitin numero *'bit'* päälle kussakin tavussa. Eniten merkitsevä bitti on bitti numero 7, ja vähiten merkitsevä bitti on bitti numero 0. (2p)

3a) Alla on funktio, joka lisää uuden int-alkion (**newval**) linkitetyn listan (**l**) loppuun, mutta se sisältää ainakin kaksi virhettä. Kerro millä riveillä virheet ovat (tai mihin kohtaan pitäisi lisätä jotain), ja kerro miten ohjelma tulisi korjata. Linkitetyssä listan alussa on "tyhjä" alkio, joka tulee sivuuttaa (kts. kuva). Älä murehdi muistin vapauttamisesta: se tehdään toisaalla ohjelmassa. Voit myös olettaa, että muistin varaus onnistuu aina. (2 p)




```
10: #include <stdlib.h>
11:
12: struct list
13: {
14:     int val;
15:     struct list *next;
16: };
17:
18: void add_to_list(struct list *l, int newval)
19: {
20:     if (!l) return;
21:     while (l->next != NULL)
22:         l=l->next;
23:     l->next = malloc(sizeof(int));
24:     l->next->val = newval;
25: }
```

Valgrind palauttaa funktiota suorittaessa seuraavaa:

```
==10788== Invalid read of size 8
==10788==    at 0x400C06: add_to_list (tentti.c:21)
==10788==    by 0x400FD6: main (tentti.c:229)
==10788==  Address 0x51ba538 is 4 bytes after a block of size 4 alloc'd
==10788==    at 0x4C28BED: malloc (vg_replace_malloc.c:263)
==10788==    by 0x400C18: add_to_list (tentti.c:23)
==10788==    by 0x400FC2: main (tentti.c:228)
==10788==
==10788== Invalid write of size 8
==10788==    at 0x400C20: add_to_list (tentti.c:23)
==10788==    by 0x400FD6: main (tentti.c:229)
==10788==  Address 0x51ba538 is 4 bytes after a block of size 4 alloc'd
==10788==    at 0x4C28BED: malloc (vg_replace_malloc.c:263)
==10788==    by 0x400C18: add_to_list (tentti.c:23)
==10788==    by 0x400FC2: main (tentti.c:228)
```

...sekä pari muuta samanlaista ilmoitusta, jotka sivuutettu tilanpuutteen vuoksi.

3b) ja c) Seuraavista funktioista puuttuu ohjelmarivejä. Valitse oikeat ohjelmarivit esitetyistä vaihtoehdoista. Kirjoita kutakin avointa riviä kohden kyseisen ohjelmarivin numero, sekä oikean vastausvaihtoehdon kirjain.

b) Funktio, joka varaa tarvittavan määrän muistia ja kopioi merkkijonon '**src**' varattuun muistipaikkaan. Funktio palauttaa osoittimen varattuun muistialueeseen. (2p)

```
20: #include <stdlib.h>
21: char *allocopy_str(const char *src)
22: {
23:     char *ptr = malloc(strlen(src) + 1);
24:     char *origptr = ptr;
25:     if (ptr) {
26:         ???
27:               ???
28:         }
29:        *ptr = 0;
29:     }
30:     return origptr;
31: }
```

Rivi 26	Rivi 27
i) while (src) {	m) ptr = src;
j) while (*src) {	n) *ptr = *src;
k) while (ptr) {	o) ptr++ = src++;
l) while (*ptr) {	p) *ptr++ = *src++;

c) Funktio 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). (2p)

```
40: int read_binary(const char *bits)
41: {
42:     int val = 0;
43:     for (int i = 7; i >= 0; i--) {
44:         ???
45:             ???
46:     }
47:     return val;
48: }
```

Rivi 44	Rivi 45
q) if (*bits++ == '1')	u) val[i] = 1;
r) if (bits++ == '1')	v) val |= 1 & i;
s) if (*bits++ == 0x1)	w) val |= (1 << i);
t) if (bits++ == 0x1)	x) *val += (1 << i);

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>

unsigned int function_A(const char *a)
{
    unsigned int c = 0;
    while (*a) {
      if (*a == '\n') c++;
      a++;
    }
    return c;
}


int function_B(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;
}


char function_C(char *a)
{
      unsigned int c[128] = { 0 };
      while (*a) {
            c[(int)*a]++;
            a++;
      }
      unsigned int s = 0;
      for (int i = 1; i < 128; i++) {
            if (c[i] > c[s])
                  s = i;
      }
      return s;
}
```

5. Seuraavassa hahmotellaan ohjelmaa, joka pitää kirjaa ajoneuvorekisteristä. Rekisteri on tallennettu linkitettynä listana. Kustakin ajoneuvosta tallennetaan rekisterinumero, jossa on enintään 7 merkkiä, sekä ajoneuvon malli, joka on vapaamuotoinen merkkijono. Ajoneuvojen taltioimiseen käytetään siis seuraavanlaista tietorakennetta:

```
struct vehicle {
    char regnro[8];
    char *model;
    struct vehicle *next;
};
```

Funktio **add_vehicle** lisää uuden ajoneuvon linkitetyn listan alkuun lukemalla käyttäjältä kaksi riviä syötettä, joista ensimmäisellä rivillä annetaan rekisterinumero (*regnro*), ja toisella rivillä annetaan ajoneuvon malli (*model*):

```
struct vehicle *add_vehicle(struct vehicle *v)
{
    struct vehicle newcar;
    newcar->next = v;
    fgets(newcar->regnro, 8, stdin);
    scanf("%s", newcar->model);
    return &newcar;
}
```

Parametri *v* osoittaa linkitetyn listan alkuun, ja voi olla myös NULL, mikäli lista on tyhjä. Funktio palauttaa osoittimen listan alkuun.

a) Funktion **add_vehicle** toteutus ei ole kovin onnistunut. Kirjoita funktio uudestaan siten että se toimii, mutta älä muuta funktion kutsurajapintaa (eli parametrien tai paluuarvon tietotyyppejä). Kannattaa esimerkiksi miettiä sitä, pitäisikö dynaamista muistia käyttää hyväksi jotenkin. Et saa muuttaa *vehicle* – tietorakennetta, mutta voit halutessasi kommentoida voisiko tietorakenteen tai funktiorajapinnan määrittelyä parantaa jollain tapaa. Voit olettaa että C-kirjaston käyttämät otsakkeet on sisällytetty toisaalla ohjelmassa. (2p)

b) Toteuta funktio `char *tell_model(struct vehicle *v, const char *r)`, joka etsii osoitteesta *v* alkavasta linkitetystä listasta rekisterinumeron *r*, ja palauttaa kyseisen ajoneuvon mallin. Mikäli rekisterinumeroa ei löydy, palautetaan NULL. (2p)

c) Toteuta funktio `void delete_all(struct vehicle *v)`, joka poistaa osoitteesta *v* alkavan linkitetyn listan ja vapauttaa kaiken listan tarvitseman muistin. (2p)

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
- `int strcmp(const char *s1, const char *s2);` palauttaa 0 jos annetut merkkijonot ovat samat

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?

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
- `char *fgets(char *s, int size, FILE *stream);` lukee enintään **size**-1 merkkiä virrasta **stream** ja kirjoittaa merkit osoitteeseen **s**.

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