AS-0.1103/4 C-ohjelmoinnin peruskurssi
Aalto-yliopiston teknillinen korkeakoulu
Tentti 12.05.2010, Raimo Nikkilä

Ohjeet

Kaikki ohjelmointitehtävät tulee toteuttaa C-kielellä hyvää ohjelmointityyliä noudattaen. Tentti arvostellaan asteikolla 0-25p ja kaikkien tehtävien painoarvo on sama (5 pistettä / tehtävä). Tentissä saa käyttää funktiolaskinta sekä tentissä jaettua C-kielen standardikirjaston minireferenssiä. Kaikki tentin tehtävät ovat tehtävissä minireferenssissä listatuilla funktioilla mutta kaikkia C-kielen standardikirjaston funktioita saa halutessaan käyttää.

Merkitse kurssikoodiksi se kurssi jota olet suorittamassa, eli joko **AS-0.1103** tai **AS-0.1104**, tätä valintaa ei voi enää muuttaa jälkikäteen. Mikäli kurssikoodia ei ole merkitty tai siinä on epäselvyyksiä, oletetaan kurssikoodiksi **AS-0.1103**.

AS-0.1103 Vastaa **viiteen** vapaavalintaiseen tehtävään.

AS-0.1104 Vastaa ainoastaan tehtäviin **1.-5.**, **ÄLÄ** vastaa tehtävään **6.**

Jaa vastauksesi konsepteille siten, että:

Tehtävät **1. ja 2.** ovat omalla konseptillaan

Tehtävät **3. ja 4.** ovat omalla konseptillaan

Tehtävät **5. ja 6.** ovat omalla konseptillaan

Tavussa (`char`) on aina 8 bittiä kaikissa tehtävissä.

Kirjoita edes opiskelijanumerosi selkeällä käsialalla tenttipapereihin.

1. Tehtävä

Selitä lyhyesti minkä toiminnon kukin alla olevista kolmesta funktiosta toteuttaa. Älä kuvaa funktion sisäistä toimintaa vaan ainoastaan se millaisen muutoksen funktio saa aikaan syötteilleen, minkä arvon funktio laskee syötteistään tai mitä funktio yleisesti ottaen tekee.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void function1(void)
5  {
6      double o, r;
7      char c;
8
9      if (scanf("%lf %c X = %lf", &o, &c, &r) != 3)
10         return;
11
12     switch(c)
13     {
14     case '+':
15         printf("X=%.2lf\n", r - o);
16         break;
17     case '-':
18         printf("X=%.2lf\n", o - r);
19         break;
20     case '*':
21         printf("X=%.2lf\n", r / o);
22         break;
23     case '/':
24         printf("X=%.2lf\n", o / r);
25         break;
26     }
27 }
28
29 #define MACRO(bs, i) ((bs)[(i) / 8] << ((i) % 8))
30
31 int function2(unsigned char* ptr, size_t n)
32 {
33     for (size_t i = 0;i < n;i++)
34         if ((MACRO(ptr, i) & 0x80) != (MACRO(ptr, n - i - 1) & 0x80));
35             return 0;
36     return 1;
37 }
38
39 void function3(FILE* f, unsigned char** p, size_t* n)
40 {
41     *n = 0;
42     *p = NULL;
43     int c;
44     while ((c = fgetc(f)) != EOF)
45     {
46         *p = realloc(*p, ++(*(n)));
47         *(*p + *n - 1) = c;
48     }
49 }
```

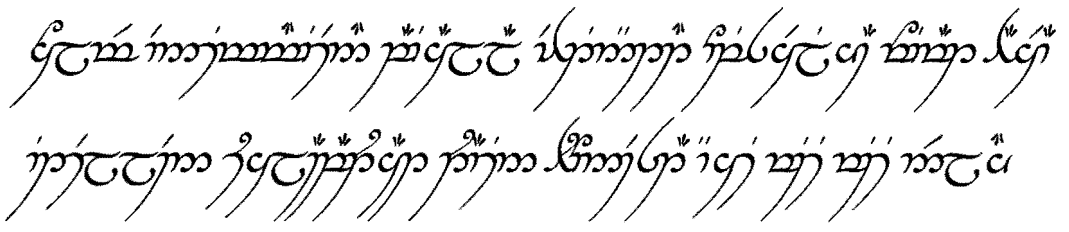
2. Tehtävä

Toteuta funktio `char* decodeString(const char* code)` joka ottaa parametrinaan koodatun merkkijonon ja purkaa sen. Koodaus koostuu kahden merkin mittaisista pareista {N,k}. Yksittäinen koodipari kuvaa `N` kappaletta `k` merkkejä, jossa $0 \leq N \leq 9$ ja `k` on jokin yksittäinen merkki. Funktio purkaa tämän koodauksen ja palauttaa puretun merkkijonon. Voit olettaa että koodattu merkkijono on aina oikean muotoinen ja että `malloc()`-funktio onnistuu aina.

Esimerkiksi koodaus ``''1a2b3c4d''`` on purettuna `abbcccdddd`.

Vihje: Merkkinä esitetyn numeron lukuarvon saat vähentämällä merkistä nollamerkin arvon.

Alla oleva teksti ei kuulu tähän tenttiin, se ole millään tavalla osa kurssisuoritusta eikä vaikuta kurssin suorittamiseen tai arvosteluun mitenkään, vaan on tuossa lähinnä ihan muuten vain.



3. Tehtävä

Mitä alla oleva ohjelma tulostaa? Vastaukseksi riittää pelkkä tuloste ilman perusteluja.
Muotoilumääre %x tulostaa kokonaisluvun heksadesimaaliesityksen.

```
1  #include <stdio.h>
2
3  unsigned char function(unsigned char p)
4  {
5      unsigned char r = 0x00;
6      for (size_t i = 0;i < 8;i++)
7          if (p & (0x80 >> i))
8              r |= (1 << i);
9      return r;
10 }
11
12 int main(void)
13 {
14     unsigned char v1 = 0x9D;
15     unsigned char v2 = 0x78;
16
17     printf("A: %x %x %x\n", v1 & v2, v1 | v2, v1 ^ v2);
18
19     printf("B: %x\n", (v1 >> 2) | (v2 << 1));
20
21     printf("C: %x\n", (unsigned char)((~v1 + 2) * 2));
22
23     printf("D: %x\n", function(v1));
24
25     printf("E: %x\n", (((v1 || v2) << 4) ^ function(0xF0)));
26 }
```

4. Tehtävä

Toteuta alla määritelty erittäin yksinkertaistettua mitokondriota kuvaava `Mc` tietotyyppi. Mitokondriot ovat kaikissa eukaryoottisissa soluissa ja eliöissä (ihminen, limasieni, ruohontupsu) esiintyviä prokaryoottista alkuperää olevia soluelimiä, jotka mm. tuottavat solulle energiaa. Toteuta kaikki neljä funktiota ja huomaa että mitokondrion "sukupuu" on lista. Kalsiummäärän puolittamisen mitoosissa voit tehdä suoraan jakolaskulla välittämättä jakojäännöksestä. Voit lisäksi olettaa että `malloc()`-funktio onnistuu aina.

```
1  #include <stddef.h>
2
3  typedef struct Mitochondrion
4  {
5    /* The genome (DNA) string of this mitochondrion */
6      char* genome;
7    /* Number of calcium molecules within the mitochondrion */
8      size_t nCa;
9    /* The parent mitochondrion from which this mitochondrion divided */
10     struct Mitochondrion* parent;
11 } Mc;
12
13 /* Creates a new mitochondrion from the given initial values and returns
14  *  the newly allocated mitochondrion.
15  * The genome of the mitochondrion is deep copied.
16  * The parent is set to NULL. */
17 Mc* McConstruct(const char* genome, size_t nCa);
18
19 /* Frees all memory allocated for a mitochondrion, including the memory
20  * allocated for all ancestor mitochondria */
21 void McDestruct(Mc* mc);
22
23 /* Performs mitosis (cell division) of a mitochondrion causing the
24  *  creation of a new mitochondrion.
25  * In mitosis, the amount of calcium molecules is shared in half between the
26  *  parent and the offspring, and the genome is deep-copied.
27  * Returns the new mitochondrion */
28 Mc* McMitosis(Mc* mc);
29
30 /* Checks for a mutation in the given strand of mitochondria.
31  * Mutation means that there is a difference in the genome of the strand,
32  *  i.e. some ancestor has a different genome than our mitochondrion.
33  * Returns a truth value indicating the presence of a mutation.
34  * Note that the parameter is not a pointer */
35 int McMutation(Mc mc);
```

5. Tehtävä

Ohessa on kokoaan kasvattavan merkkijonotaulukon, `strArray` tietotyypin, toteutus. Valitettavasti tämä toteutus on parhaimmillaankin toisluokkainen, sillä siinä on muutamia selkeitä virheitä. Etsi nämä virheet ja kuvaile **lyhyesti** miten korjaisit ne. `malloc()`- ja `realloc()`-funktioiden oletetaan onnistuvan aina. Pääohjelmassa ei ole virheitä ja muun ohjelman tulisi toimia pääohjelman mukaan.

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4
5  typedef char** strArray;
6
7  /* Creates an empty strArray */
8  strArray strArrayCreate(void)
9  {
10     return NULL;
11 }
12
13 /* Adds a deep-copy of a string to the end of an strArray */
14 strArray strArrayAdd(strArray arr, const char* str)
15 {
16     size_t s;
17     for (s = 0;arr[s];s++);
18     arr = realloc(arr, sizeof(*arr) * (s + 2));
19     arr[s] = malloc(strlen(str + 1));
20     strcpy(arr[s], str);
21     return arr;
22 }
23
24 /* Removes the first incidence of a string from an strArray */
25 void strArrayDelete(strArray arr, const char* str)
26 {
27     strArray tmp;
28     for (tmp = arr;*tmp;tmp++)
29         if (str == *tmp)
30             break;
31     if (!*tmp)
32         return;
33     do {
34         *tmp = tmp[1];
35     } while(*tmp++);
36 }
37
38 /* Calculates the number of elements in an strArray */
39 size_t strArrayLen(const strArray arr)
40 {
41     strArray tmp = arr;
42     while (*tmp)
43         tmp++;
44     return (tmp - arr) * sizeof(char*);
45 }
46
47 /* Just a sample main() - no errors here so don't bother fixing it */
48 int main(void)
49 {
50     strArray arr = strArrayCreate();
51     char buffer[512];
52     while (fgets(buffer, 512, stdin) && *buffer)
53         arr = strArrayAdd(arr, buffer);
54     size_t len = strArrayLen(arr);
55     printf("Read %zu strings\n", len);
56     for (size_t n = 0;n < len;n++)
57         strArrayDelete(arr, *arr);
58     free(arr);
59 }
```

6. Tehtävä

Tarkastellaan funktioita jotka ottavat parametrinaan kokonaisluvun (`int`) ja palauttavat liukuluvun (`double`), eli funktioita $f : \mathbb{Z} \to \mathbb{R}$, rajatulla parametrien lukualueella $[a, b]$, jossa $\forall a, b : \ a < b$.

Tällä alueella funktion sanotaan olevan aidosti kasvava, jos pätee $\forall n, n \in [a, b[\ \ f(n) < f(n+1)$, eli funktion arvot kasvavat aina parametrin kasvaessa. Lisäksi, funktion sanotaan olevan toisen funktion rajoittama jos $\forall n, \ n \in [a, b] \ \ f(n) < g(n)$, eli jos funktion arvot määrätyllä alueella ovat aina pienempiä kuin toisen funktion arvot.

Toteuta funktiot `monotonicallyIncreasing()` ja `boundedFunction()` jotka molemmat ottavat ensimmäisinä parametreina lukualeen alun ja lopun kokonaislukuina (`int`). Funktio `monotonicallyIncreasing()` ottaa lisäksi yhden edellämainittua tyyppiä olevan funktion osoitteen ja testaa onko funktio monotonisesti kasvava annetulla lukualueella ja palauttaa sitä vastaavan totuusarvon. Funktio `boundedFunction()` ottaa lukualueen lisäksi parametreina kahden funktion osoitteet ja testaa onko ensimmäisenä annettu funktio toisena annetun funktion rajoittama ja palauttaa sitä vastaavan totuusarvon.

ctype.h

```
int isalnum(int ch); int isalpha(int ch); int isdigit(int ch);
int islower(int ch); int ispunct(int ch); int isspace(int ch);
int isupper(int ch); int tolower(int ch); int toupper(int ch);
```

math.h

```
double pow(double base, double exponent);
double sqrt(double value);
```

stdio.h

```
int printf(const char* format, ...);
int fprintf(FILE *stream, const char* format, ...);
int sprintf(char *str, const char* format, ...);
int scanf(const char* format, ...);
int fclose(FILE *fp);
FILE *fopen(const char *path, const char *mode);
int fscanf(FILE *stream, const char* format, ...);
int sscanf(const char *str, const char* format, ...);
int fgetc(FILE *stream);
char *fgets(char *str, int size, FILE *stream);
int fputc(int ch, FILE *stream);
int getchar(void);
int putchar(int ch);
```

stdlib.h

```
void free(void *ptr);
void* malloc(size_t size);
void* calloc(size_t nmemb, size_t size);
void* realloc(void* ptr, size_t size);
void abort(void);
void exit(int);
void qsort(void* base, size_t nmemb, size_t size, int (*cmp)(const void*, const void*));
int abs(int val);
```

string.h

```
char* strcat(char *dest, const char *src);

char* strchr(const char *str, int ch);

int strcmp(const char *str1, const char *str2);

char* strcpy(char *dest, const char *src);

size_t strlen(const char *str);

char *strstr(const char *haystack, const char *needle);

size_t strxfrm(char *dest, const char *src, size_t n);

void* memset(void *s, int c, size_t n);
```