Tentamen i Programutveckling LEU481

Joachim von Hacht

Datum: 2020-06-04

Tid: 08.30-12.30

Hjälpmedel: Allt.

Betygsgränser:

U: 0-19

3: 20-29

4: 30-39

5: 40-50 (max 50)

Lärare: Joachim von Hacht tel. 772 10 03. Alla frågor skickas via mail: hajo@chalmers.se

Granskning: Anslås på kurssida.

Instruktioner:

- ALLA SVAR SKALL SKRIVAS I RESPEKTIVE FRÅGE-FIL: f1.c, f2.c, f3.c, ... f8.c. Hela tentan (projektet) skall zippas och lämnas i Canvas.
- Vi utgår från att användaren alltid skriver rätt och/eller gör rätt (d.v.s ingen felhantering behövs). Om felhantering skall ingå anges detta specifikt.
- Lösningarna måste klara de fall som anges *samt fall som är principiellt lika.* Lösningar som bara klarar exemplen räcker *inte.* Överkomplicerade lösningar kan ge poängavdrag.
- Färdiga funktioner som får användas anges i uppgiften. De finns beskrivna i Appendix. Anges inget får man alltid använda de grundläggande språkliga konstruktionerna och arrayer, matriser, strukturer och egna funktioner.

LYCKA TILL...

1. Koden nedan fungerar inte. Varför? Ge en kort förklaring (skriv som kommentar direkt i koden). 2p

```
char *str = "abcdef";
printf("%c\n", str[0]);
str[1] = 'X';
printf("%s\n", str);
```

2. Skriv ett program som använder formeln för linjär interpolation för att, givet ett x-värde, beräkna ett y-värde. 3p

   Formel för linjär interpolation: $y = y1 + \frac{(x-x_1)(y_2-y_1)}{(x_2-x_1)}$

   (x_1, y_1) och (x_2, y_2) är två givna punkter och x ligger mellan x_1 och x_2 . Exempel:

```
Skriv in x1 och y1 > 1 2
Skriv in  x2 och y2 > 3 4
Skriv in x för sökt y värde > 2
Linjär interpolation ger y = 3.000000
```

3. Skriv ett program som tränar delbarhet (för mellanstadieelever). Programmet genererar två slumptal $1 \leq n_1 \leq 20$ och $1 \leq n_2 \leq 9$ och frågar om $\frac{n_1}{n_2}$ är jämnt delbart. Användaren svarar med 1 för sant och 0 för falskt. Programmet ställer 10 frågor och håller rätt på antal rätta svar och totala antalet svar. Detta skrivs ut då programmet avslutas. Om man vill avbryta innan alla frågor är ställda svarar man -1. Om så skrivs bara "Avbrutet" ut. Exempel: 6p

```
Träna delbarhet. Ange 0 för falskt och 1 för sant, avsluta med -1
Är 9 delbart med 3? 1
Rätt på fråga 1
Är 8 delbart med 2? 1
Rätt på fråga 2
Är 15 delbart med 9? 0
Rätt på fråga 3
Är 18 delbart med 10? 1
Fel på fråga 4
Är 12 delbart med 1? 1
Rätt på fråga 5
Är 8 delbart med 5? 1
Fel på fråga 6
Är 2 delbart med 6? 0
Rätt på fråga 7
Är 20 delbart med 7? 0
Rätt på fråga 8
Är 14 delbart med 6? 0
Rätt på fråga 9
Är 1 delbart med 5? 0
Rätt på fråga 10
Du hade 8 rätt av 10 frågor
```

TIPS: Ibland behöver man läsa bort ett kvarvarande "Enter"-tecken ('\n'). Görs med ett anrop till getchar()

4. Skriv en funktion, void partial_sums(int p_sums[], const int arr[], int size), som givet en heltalsarray returnerar en array med alla partialsummor i arrayen. Vi antar att båda arrayerna är lika långa. Exempel: 4p

```
arr                p_sums (resultatet)
[1, 1, 2]          [1, 2, 4]     (1, 1+1, 1+1+2)
[1, 2, 3, 4, 5]    [1, 3, 6, 10, 15] (1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5)
[0, 0, 0]          [0, 0, 0]
[1, 2,-3]          [1, 3, 0]
[9]                [9]
```

5. Skriv en metod, void expand_string(char *expanded, const char *str) som "expanderar" en given sträng. Alla metoder i appendix är tillåtna. Expansionen sker enligt följande: 6p

```
str                expanded
-------            ----------
"2(@f)"            "@f@f"
"3(ab)2(?)1(xyz)"  "abababab??xyz"
"1(21)2(((()"      "21(((("
```

D.v.s strängen består av en eller flera grupper där varje grupp inleds med en siffra (0-9) och därefter en parentes med ett antal tecken. Strängen expanderas så att tecknen i parentesen upprepas lika många gånger som siffran anger. De omslutande parenteser tas bort. I parentesen förekommer inget ')'-tecken. Den expanderande strängen är kortare än 100 tecken.

TIPS: För att få ett sifferteckens numeriska värde kan man subtrahera med '0' (tecknet för 0, ASCII-kod 48) t.ex. '1'-'0' ger heltalet 1.

6. Rita bilder som visar variabler, värden och pekare (pilar) då programmet kört raderna // Before respektive // After i koden nedan. 8p

```
int m[][2] = {
   {1, 2},
   {3, 4}};
int a[] = {60, 70};
int *p1 = m[1];
int *p2 = a;          // Before
p2 = do_it(p1, p2); // Call
                     // After

int *do_it(int *a, int *b) {
    a--;
    b++;
    int tmp = *a;
    *a = *b;
    *b = tmp;
    return a + 2;
}
```

7. En "växande" matris är en matris där värdena i både rader och kolumner ökar för varje element. Exempel: 7p

```
[1, 2, 3     // Växande
 4, 5, 6
 7, 8, 9]

[1, 2, 3    // Inte växande
 2, 3, 6    // Rader ok men inte kolumn 2
 3, 4, 5]

[1, 4, 3   // Inte växande
 2, 6, 7]  // Kolumner ok men inte rad 1

[1, 3, 5   // Växande
 2, 4, 6
 3, 5, 7]
```

   Skriv en funktion, bool is_increasing(int n_rows, int n_cols, int m[][n_cols]), som givet en matris av heltal avgör om matrisen är växande. Om så returneras true annars false. För full poäng krävs funktionell nedbrytning.

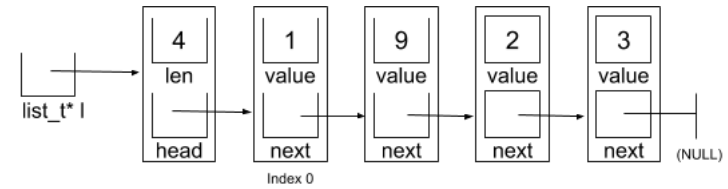
8. Vi skall skriva ett program som använder linjer i planet. 6p

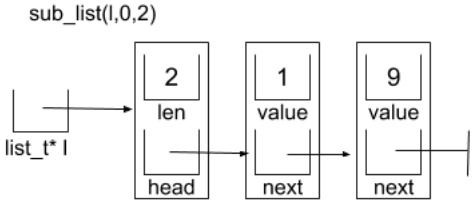
   a) Skapa en struktur för punkter i planet. Använd denna för att skapa en struktur för linjer där linjen representeras som två punkter, startpunkt och slutpunkt. Använd typedef för båda strukturerna för att få kortare namn. Döp den till point2D_t och line2D_t.

   b) Skapa en funktion som givet en linje returnerar längden av linjen. Funktionen sqrt är tillåten.

   c) Visa hur du i main-funktionen skapar och initierar en variabel för en linje samt anropar funktionen för längden (och sparar resultatet).

9. Antag att vi har en länkad lista enligt bilden och koden nedan. Skriv en funktion, list_t *sub_list(list_t *list, int from_index, int to_index) som givet en lista returnerar en sublista från from_index (inklusive) till to_index (exklusive). Vi antar att användaren skriver in rimliga index. Index är 0-baserat som för vanliga arrayer. Du får använda de färdiga funktionerna list_new och list_append, se nedan (kompakt skriva p.g.a. platsbrist). 8p






```
typedef struct Node {    // Nodes in list
   int value;
   struct Node *next;
} node_t;
typedef struct List {    // The list
   int len;
   node_t *head;
} list_t;
list_t *list_new() {
   list_t *list = malloc(sizeof(list_t));
   list->head = NULL; list->len = 0;
}
list_t *list_append(list_t *list, int value) {
   node_t *n = malloc(sizeof(node_t)); n->value = value; n->next = NULL;
   if (list->len == 0) {
      list->head = n;
   } else {
      node_t *pos = list->head;
      while (pos->next != NULL) {
         pos = pos->next;
      }
      pos->next = n;
   } list->len++; }
```

APPENDIX

Följande makron/funktioner får användas om så anges i uppgiften.

```
Funktioner ur math.h
double sqrt( double n)  // Roten ur n

Funktioner (makron) ur ctype.h. Alla funktioner tar ett tecken som
parameter (trots att typen är int)
int isalpha(int ch);   // Sant om ch är bokstav
int isdigit(int ch);   // Sant om ch är siffra
int islower(int ch);   // Sant om ch är liten bokstav
int ispunct(int ch);   // Sant om ch är något av .,;:?! (antar vi)
int isupper(int ch);   // Sant om ch är stor bokstav
int tolower(int ch);   // Omvandlar till liten bokstav
int toupper(int ch);   // Omvandlar till stor bokstav

Funktioner ur string.h
// Längden av strängen str (\0 inte medräknad). size_t är ett teckenlöst heltal
size_t strlen(const char *str)

// Ger < 0 om str1 < str2, 0 om lika, > 0 om str1 > str2
// Jämförelsen gör tecken för tecken vänster till höger m.h.a. ASCII koder.
// Ej svenska tecken.
int strcmp(const char *str1, const char *str2)

// Kopierar src (inklusive 0) till dest. Returvärdet är pekare till dest.
// Ingen kontroll av plats. Minnespositionerna får inte överlappa.
char *strcpy(char *dest, const char *src)
// Som ovan men kopierar num tecken
char *strncpy(char *dest, const char *src, size_t num )

// Lägger till src från slutet på dest (börjar på platsen för dest's \0).
// Ingen kontroll av plats. Minnespositionerna får inte överlappa.
char *strcat(char *dest, const char *src)
// Som ovan men kopierar num tecken
char *strncat(char *dest, const char *src, size_t num)

// Söker efter tecknet ch i str. Pekare till tecken om det finns annara NULL
char* strchr(const char* str, char ch)

Satsen srand((unsigned) time(NULL)) och funtionen rand() får alltid användas.
```