Technische Universität Hamburg-Harburg
Institut für Zuverlässiges Rechnen
Prof. Dr. S.M. Rump

SS 2012

Lösungen Klausur Prozedurale Programmierung

24. September 2012

Sie haben 90 Minuten Zeit zum Bearbeiten der Klausur.

Tragen Sie bitte zunächst Ihren Namen, Ihren Vornamen, Ihre Matrikelnummer, Ihr Studienfach und Ihr Studiensemester in **DRUCKSCHRIFT** in die folgenden jeweils dafür vorgesehenen Felder ein.

Name:														
Vorname:														
Matr.-Nr.:								Fach				Sem.		

Bei jeder der fünf Aufgaben können 12 Punkte erreicht werden, insgesamt also 60 Punkte. Es sind alle Aufgaben zu bearbeiten. Zum Bestehen der Klausur sind 30 Punkte erforderlich.

Aufg.	Punkte	Korr.
1		
2		
3		
4		
5		
$\sum$		

Note	

Zur Beachtung:
Schreiben Sie bitte weder mit Bleistift noch mit Rotstift. Es sind keinerlei Hilfsmittel in dieser Klausur zugelassen! Das Schreiben vor dem Startsignal und auch das Schreiben nach dem Endsignal führt ohne weitere Warnung sofort zur Ungültigkeit der Klausur. Dies gilt auch für das Schreiben von Namen und Matrikelnummer nach dem Endsignal. Vergessen Sie nicht, den „Vorbehalt“ zu unterschreiben.
Hinweis zum Programmieren:
Programmieren Sie die Aufgaben in ANSI-C.

Vorbehalt
Ich bin darüber belehrt worden, dass die von mir zu erbringende Prüfungsleistung nur dann bewertet wird, wenn die Nachprüfung durch das Zentrale Prüfungsamt der TUHH meine offizielle Zulassung vor Beginn der Prüfung ergibt.

_____

(Datum, Unterschrift)

Aufgabe 1 (12 Punkte)

Folgendes Programm demonstriert eine einfache Methode zur Datenverschlüsselung. Dem Programmierer sind 12 syntaktische und semantische Fehler unterlaufen. Finden Sie diese!

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #define MASKSHIFT(a,b,c) (((a)>>(b)) + ((a)<<((c)-(b))))
5  void crypt(FILE *FileIn, FILE *FileOut, char pwd[]) {
6    const int csize = sizeof(char);
7    char chr=\0;
8    int i=0, j=0, k=0, pwdlen=0;
9    pwdlen=strlen(pwd);
10   if (pwdlen<1) { printf("Fehler: Passwortuebergabe\n"); exit(1); }
11   i=-1; j=2;
12   while (fscanf(FileIn,"%c",chr)!=EOF) {
13     i=(i+1)%pwdlen;
14     j=(j+i+1)%pwdlen;
15     k=(i+j) mod csize;
16     chr=chr^(MASKSHIFT(pwd[i],k,csize))^pwd[j];
17     fprintf(FileOut,"%c",chr);
18   }
19 }
20 int main(int argc, char argv[]) {
21   char *filename; *pwd;
22   FILE *pFileIn, *pFileOut;
23   if (argc<3) { printf("Fehler beim Aufruf\n"); exit(1); }
24   pwd=argv[1];
25   pFileIn=fopen(argv[2],"r");
26   filename=(char)malloc((4+strlen(argv[2]))*sizeof(char));
27   if (filename) { printf("Fehler: Speicherallokation\n"); exit(1); }
28   strcpy(filename,"enc_");
29   strcat(filename,argv[2]);
30   pFileOut=fopen(filename,"w");
31   if (pFileIn==NULL || pFileOut==NULL) {
32     printf("Fehler: fopen\n"); exit(1);
33   }
34   crypt(pFileIn,pFileOut,pwd);
35   fclose(pFileIn); fclose(pFileOut);
36   pFileIn=fopen(filename,"readonly");
37   if (argc>3) { *filename=*(argv+3); }
38   else
39     strcpy(filename,"dec_");
40     strcat(filename,argv[2]);
41   }
42   pFileOut=fopen(filename,'w');
43   if(pFileIn==NULL || pFileOut==NULL) {
44     printf("Fehler: fopen",\n); exit(1);
45   }
46   crypt(pFileIn,pFileOut,pwd);
47   fclose(pFileIn); fclose(pFileOut);
48   return(0);
49 }
```

Zeile	Notieren Sie die korrigierten Zeilen (maximal 12) hier!
7	`char chr='\0';`
12	`while (fscanf(FileIn,"%c",&chr)!=EOF) {`
15	`k=(i+j)%csize;`
20	`int main(int argc, char *argv[]) {`
21	`char *filename, *pwd;`
26	`filename=(char*)malloc((4+strlen(argv[2]))*sizeof(char));`
27	`if (filename==NULL) { printf("Fehler: Speicherallokation\n"); exit`
36	`pFileIn=fopen(filename,"r");`
37	`if (argc>3) { filename=*(argv+3); }`
38	`else {`
42	`pFileOut=fopen(filename,"w");`
44	`printf("Fehler: fopen\n"); exit(1);`

Aufgabe 2 (12 Punkte)

Durch ein Programm sollen n Länder einer Landkarte so eingefärbt werden, dass benachbarte Länder unterschiedliche Farben haben. Dafür stehen insgesamt vier Farben zur Verfügung, welche durch die Ziffern 1 bis 4 kodiert sind. Folgendes darf man als gegeben annehmen:

(1) n ist global definiert.

(2) Zwei Länder i, j sind genau dann benachbart, wenn $M[i][j] = 1$ gilt, wobei M global definiert ist.

(3) C ist ein globales Array der Länge n, in welchem letztendlich die Farbziffern der Länder gespeichert werden sollen. Zu Beginn ist es mit Nullen initialisiert.

Die Färbung soll durch eine **rekursive** Funktion `color` erfolgen, welche als Eingabeparameter eine Farbziffer c und eine Ländernummer $i \in \{0, \ldots, n-1\}$ erhält. Rückgabewerte sind entweder 1: "erfolgreich" oder -1: "nicht erfolgreich". Folgender Algorithmus soll implementiert werden:

(a) Falls $c > 4$, so war der Färbevorgang nicht erfolgreich.

(b) Falls $i = n$, so wurden bereits alle Länder erfolgreich gefärbt.

(c) Falls $c \leq 4$ und $i < n$, untersuche man, ob alle Nachbarn von i eine von c verschiedene Färbung haben. Wenn nicht, versuche man es rekursiv mit der nächsten Farbe und breche die Funktion mit dem Rückgabewert dieser Rekursion ab. Andernfalls bekommt Land i die Farbe c und es erfolgt ein geeigneter rekursiver Aufruf für das nächste Land. Endet dieser erfolgreich, so war c eine richtige Farbwahl, andernfalls wird Land i wieder entfärbt und die Funktion endet erfolglos.

Notieren Sie die Funktion `color` hier.

```
1   int color(int c,int i){                                  /* 2 Punkte */
2     int j=0;
3     if(i==n) return 1;                                     /* 1 Punkt  */
4     if(c>4) return -1;                                     /* 1 Punkt  */
5     for(j=0;j<i;j++)
6       if(M[i][j]==1 && C[j]==c) return color(c+1,i);       /* 3 Punkte */
7     C[i]=c;                                                /* 1 Punkt  */
8     if(color(1,i+1)==1) return 1;                          /* 2 Punkte */
9     C[i]=0;                                                /* 1 Punkt  */
10    return -1;
11  }
```

Wie lautet der Funktionsaufruf im Hauptprogramm ? `color(1,0)` `/* 1 Punkt */`

Aufgabe 3 (12 Punkte)

a) Schreiben Sie eine Funktion `lut4log`, die eine Lookup-Tabelle für den Logarithmus erzeugt. Der Wertebereich wird durch drei Eingabeparameter festgelegt, zwei `double`-Variablen, `start` und `end`, sowie die `int`-Variable n, welche die Anzahl der äquidistanten Tabellenelemente angibt. Die Funktion soll einen `double`-Zeiger auf das erstellte LUT-Array zurückgeben. Achten Sie auch darauf, falsche Eingabewerte und Fehler bei der Speicherreservierung abzufangen! (4 Punkte)

Hinweis: Das i-te Element der LUT ergibt sich zu: $\log(start + i\ \frac{end-start}{n-1})$, $i \in \{0, \ldots, n-1\}$.

```
1   #include <stdlib.h>
2   #include <math.h>
3
4   double* lut4log(double start, double end, int n) {          /* 1 Punkt */
5     int i=0; double *a=NULL;
6
7     if (n<2 || end<=0 || start<=0) return NULL;               /* 1 Punkt */
8     a=(double *) malloc(sizeof(double)*n);                    /* 1 Punkt */
9     if (a==NULL) return a;
10    for (i=0; i<n; ++i)  a[i]=log(i*(end-start)/(n-1)+start); /* 1 Punkt */
11    return a;
12  }
```

b) Es sei `pLUT` ein Zeiger auf eine Lookup-Tabelle, wie sie zum Beispiel mit der Funktion aus `a)` erzeugt werden kann. Ferner sei `S` eine Struktur, die neben anderen Komponenten eben diesen Zeiger enthält. Die Variable `A` ist als ein Feld aus Instanzen dieser Struktur deklariert worden. Geben Sie vier verschiedene Zugriffsmöglichkeiten auf einen bestimmten LUT-Eintrag an, die äquivalent zum folgenden Aufruf sind: `A[4].pLUT[0]` ! (4 Punkte)

```
(A+4)->pLUT[0]   *((A+4)->pLUT)   *(A[4].pLUT)   (*(A+4)).pLUT[0]   *((*(A+4)).pLUT)
```

c) Welche Ausgabe erzeugt das folgende Programm? (4 Punkte)

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   int main(void) {
5     int i=0, j=0, n=15;
6     char temp='\0';
7     char term[]="{beg}12+x=33{end}";
8     for(i=0, j=n-1; i<n; i++,j--) {
9       temp=term[i];
10      term[i]=term[j];
11      term[j]=temp;
12    }                                                  /* Ausgabe:    */
13    for(i=3; i<7; i++) {                               /*             */
14      j=9-(i%5);                                       /* out[6] =  2 */
15      printf("out[%d] = %c\n",j,term[j]);              /* out[5] =  1 */
16    }                                                  /* out[9] =  = */
17    return 0;                                          /* out[8] =  x */
18  }
```

Aufgabe 4 (12 Punkte)

a) Stellen Sie die Dezimalzahl -1.59375 als Fließkommazahl im 32 Bit IEEE 754 Standard dar. (3 Punkte)

Vorzeichenbit:

1

Exponent: (mit dem Shift 127)

0	1	1	1	1	1	1	1

Mantisse: (Vergessen Sie nicht die implizite Eins!)

1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b) Stellen Sie in der ersten Zeile die Dezimalzahl -432 im 2-er Komplement (`short int`) binär dar. Die zweite Zeile enthält bereits eine solche Binärdarstellung. Geben Sie links den entsprechenden Dezimalwert an. Addieren Sie abschließend in der letzten Zeile beide Zahlen binär. (3 Punkte)

-432 :	1	1	1	1	1	1	1	0	0	1	0	1	0	0	0	0
-456 :	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0
$\sum$:	1	1	1	1	1	1	0	0	1	0	0	0	1	0	0	0

c) Welche Ausgaben erzeugt die folgende Code-Sequenz ? Hinweis: `%hd` ist das Ausgabeformat für `short int`.

```
1    short int c=30000, d=c++;
2    char str[10]="abc\0DEF", *pstr=str;
3    printf("\n%hd",c+d);                          /*1.Ausgabe: -5535      */
4    printf("\n%s%s",pstr,pstr+4);                 /*2.Ausgabe: abcDEF     */
5    printf("\n%c",'b'+'C'-'a');                   /*3.Ausgabe: D          */
```

Begründen Sie Ihre Antwort: (3 Punkte)

1. $c + d = 30000 + 30001 = 60001 = 32767 + 1 + 27233$ wrap around: $-32768 + 27233 = -5535$
2. Wegen `str[3]=0` liefert `printf("%s",pstr,)` das Ergebnis `abc`. Anschließend Ausgabe des restlichen Strings ab `str[4]='D'`, i.e. `DEF`.
3. `'b'+'C'-'a'='b'+('C'-'c')+('c'-'a') = 'B'+2 = 'D'`

d) Schreiben Sie eine Funktion `void str_sort(char s[])`, welche die Buchstaben eines Strings gemäß ASCII in aufsteigender Reihenfolge sortiert. Sie können `strlen(s)` zur Bestimmung der Länge des Eingabestrings `s` verwenden. **Beispiel:** `"PC-Spiel"` wird zu `"-CPSeilp"`. (3 Punkte)

```
1   void str_sort(char s[]){
2     int i,j;
3     char c;
4     for(i=strlen(s)-1;i>0;i--)
5       for(j=0;j<i;j++)
6         if(*(s+j)>*(s+j+1)){c= *(s+j);*(s+j)=*(s+j+1);*(s+j+1)=c;}
7     return;
8   }
```

Aufgabe 5 (12 Punkte)

a) Ein Unternehmen will die Personalnummern seiner Mitarbeiter in einer verketteten Liste verwalten. Definieren Sie hierfür eine Struktur `sEmpList`, welche die Komponenten `PersNr` für die ganzzahlige Personalnummer und `next` für den Zeiger auf den Nachfolger besitzt.

(3 Punkte)

```
100  struct sEmpList{
101    int PersNr;
102    struct sEmpList *next;
103  }; /*sNameList*/
```

b) Schreiben Sie eine Funktion `NewEmployee`, die eine neue Personalnummer am Anfang einer Liste mit Struktur aus a) einfügt. Eingabeparameter sind ein Zeiger `E` auf den Listenanfang und die neue Personalnummer `PersNr`. Rückgabewert soll ein Zeiger auf den Anfang der erweiterten Liste sein.

(3 Punkte)

```
104  struct sEmpList *NewEmployee(struct sEmpList *E, int PersNr){
105    struct sEmpList* F;
106    F = (struct sEmpList*) malloc (sizeof(struct sEmpList));
107    F->PersNr=PersNr;
108    F->next=E;  /*neues Element an den Anfang setzen*/
109    return(F);
110  }
```

c) Eine Methode um Zugriffszeiten auf häufig angefragte Personalnummern zu verringern besteht bei sogenannten adaptiven Listen darin, bei jedem Zugriff auf ein Element dieses mit seinem Vorgänger zu vertauschen. Schreiben Sie eine Funktion `swap`, die ein Listenelement mit seinem Vorgänger vertauscht. Sie besitze die gleichen Eingabe- und Rückgabeparameter wie die Funktion `NewEmployee` aus b).

(6 Punkte)

```
111  struct sEmpList* swap(struct sEmpList* E,int PersNr){
112    struct sEmpList *F; struct sEmpList *G;
113    if(E!=NULL && E->next!=NULL){ /*Für leere oder 1-elementige Listen ist nichts zu tun.*/
114      F=E;
115      if(E->next->PersNr == PersNr){
116      /* Das gesuchte Listenelement steht an zweiter Stelle, wird daher bei Vertauschung
117         erstes Listenelement und damit zum Listenzeiger.*/
118        E=E->next;
119        F->next =E->next;
120        E->next = F;
121      }
122      else{
123        while(F->next->next != NULL && F->next->next->PersNr != PersNr){F=F->next;}
124        if(F->next->next != NULL){ /*Dann gilt F->next->next->PersNr == PersNr*/
125          G=F->next;
126          F->next=F->next->next;
127          G->next=F->next->next;
128          F->next->next =G;
129        }
130      }
131    }
132    return(E);
133  }
```