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

SS 2011

Musterklausur Prozedurale Programmierung

25. Juli 2011

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 soll aus einer im Kommandofenster übergebnenen C-Quelldatei alle Kommentare entfernen und den veränderten Inhalt in einer temporären Datei speichern. Dem Programmierer sind 12 syntaktische und semantische Fehler unterlaufen. Finden Sie die Fehler.

```
1  #include <stdio.h>
2  int main(int argc, char *argv[]) {
3      char chr, rcm = 0, nstr = 1;
4      FILE *file, *tmp;
5  /*    if (argc < 2 || argc > 2) {
6          printf("Fehler: keine Eingabdatei\n");
7          return -1;
8      }    */
9      file = fopen("A3.c","r");
10     tmp = fopen("./tmp.c","w");
11     if (file == NULL) {
12         printf("Konnte die Datei %s nicht öffnen\n",argv[1]);
13         return -1;
14     }
15     if (tmp == NULL) {
16         printf("Konnte temporäre Datei nicht öffnen\n");
17         return -1;
18     }
19     while (fscanf(file,"%c",&chr) != EOF) {
20         switch (chr)
21         {
22             case '\"':
23                 nstr = !nstr;
24                 printf("case1\n");
25             case '/':
26                 if (fscanf(file,"%c",&chr) != EOF && chr == '*' && nstr)
27                     rcm = 1;
28                 else
29                     fprintf(tmp,"%c",'/');
30                 printf("case2\n");
31                 break;
32             case '*':
33                 if (fscanf(file,"%c",&chr) != EOF && chr == '/' && nstr) {
34                     rcm = 0;
35                     chr = ' ';
36                 }
37                 else
38                     fprintf(tmp,"%c",'*');
39                 printf("case3\n");
40                 break;
41             default: ;
42         }
43         if (!rcm) {
44             fprintf(tmp,"%c",chr);
45         }
46     }
47     fclose(file); fclose(tmp);
48     return 0;
49 }
```

Zeile	Notieren Sie die korrigierten Zeilen (maximal 12) hier !
2	`int main(int argc, char *argv[]) {`
3	`char chr, rcm = 0, nstr = 1;`
6	`printf("Fehler: keine Eingabdatei\n");`
9	`file = fopen(argv[1],"r");`
10	`tmp = fopen("./tmp.c","w");`
15	`if (tmp == NULL) {`
19	`while (fscanf(file,"%c",&chr) != EOF) {`
24	`break;`
31	`case '*':`
39	`default: ;`
42	`fprintf(tmp,"%c",chr);`
45	`fclose(file); fclose(tmp);`

Aufgabe 2 (12 Punkte)

Schreiben Sie eine rekursive Funktion `instring`, die zwei Strings `str1`, `str2` als einzige Eingabeparameter besitzt und testet, ob `str1` ein Teilstring von `str2` ist. In diesem Fall soll die Anfangsposition von `str1` in `str2` zurückgegeben werden. Wenn `str1` kein Teilstring von `str2` ist, soll -1 zurückgegeben werden.

Beispiel: Der Aufruf `instring("bahn","Eisenbahn")` soll das Ergebnis 5 liefern, während der Aufruf `instring("Bahn","Eisenbahn")` zum Ergebnis -1 führen soll.

Hinweise:

1. Ein leerer String `""` ist Anfangsstring eines jeden anderen Strings.
2. Ein nichtleerer String ist kein Teilstring eines leeren Strings.
3. Für einen nichtleeren String `str` bezeichne `str_+` den Reststring von `str` nach dem ersten Buchstaben. Wenn nun die Anfangsbuchstaben von `str1` und `str2` übereinstimmen, so ist `str1` genau dann ein Anfangsstring von `str2`, wenn `str1_+` ein Anfangsstring von `str2_+` ist.
4. Wenn andererseits die Anfangsbuchstaben von `str1` und `str2` nicht übereinstimmen und `str1` ein Teilstring von `str2_+` ist, so lässt sich die Startposition von `str1` in `str2` leicht aus der Startposition von `str1` in `str2_+` ermitteln. Falls `str1` kein Teilstring von `str2_+` ist, so ist das Ergebnis ebenfalls klar.

Bemerkung: Eine unkomplizierte Lösung der Aufgabe ist in 9 Zeilen möglich.

```
int instring(char *str1, char *str2){ /* 2 Punkte  */
  int r;
  if(*str1==0) return(0);                                          /* 2 Punkte  */
  if(*str2==0) return -1;                                          /* 2 Punkte  */
  if(*str1==*str2 && instring(str1+1,str2+1)==0) return(0);        /* 3 Punkte  */
  r=instring(str1,str2+1);                                         /* 3 Punkte */
  if(r!=-1) return(r+1);
  else return(-1);
}
```

Aufgabe 3 (12 Punkte)

a) Schreiben Sie eine `void`-Funktion, die als Parameter ein `char`-Array und einen Integer-Wert, der die Länge des `char`-Arrays angibt, besitzt. Die Funktion soll Inhalt und Adresse jedes Array-Elementes übersichtlich ausgeben.

```
1  #include <stdio.h>
2
3  void printarray(char a[], int n) {
4      int i;
5      for (i = 0; i < n; ++i)
6          printf("a[%-5d] = %-5x   &a[%-5d] = %p\n",i,a[i],i,a+i);
7      return;
8  }
```

(3 Punkte)

b) Die Variable `A` sei folgendermaßen deklariert worden: `char A[5][10]`. Geben Sie drei verschiedene, äquivalente Zugriffsmöglichkeiten auf das Element aus der dritten Zeile und vierten Spalte von `A` an.

```
A[2][3]      *(A[2] + 3)      *(*(A + 2) + 3)
```

(3 Punkte)

c) Welche Ausgabe erzeugt das folgende Programm ?

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #define N 5
4
5  void setarray(int *a, int n) {
6      int b[] =  {5,4,3,2,1}, i;
7      for (i = 0; i < n; ++i)
8          a[i] = b[i+3];
9  }
10
11 int main(void) {
12     int a[N] = {1,2,3,4,5}, i;
13     setarray(a+*(a+2),N-a[2]);
14     for(i = *(a+1); i < N; ++i)
15         printf("a[%d] = %d\n",i,a[i]);
16     return 0;
17 }
```

Ausgabe:

```
a[2] = 3
a[3] = 2
a[4] = 1
```

(3 Punkte)

d)

i) Eine Variable `X` sei vom Typ einer nicht näher spezifizierten Struktur `S`, die eine Komponente `lii` vom Datentyp `long int` besitzt. Geben Sie eine Anweisung zum Einlesen eines ganzzahligen Wertes in die Komponente `lii` von `X` an.

```
scanf("%ld",&(X.lii));
```

(1 Punkt)

ii) Wie lautet die Anweisung, wenn `X` ein Zeiger auf `S` ist und bereits Speicher reserviert wurde ?

```
scanf("%ld",&(X->lii));   oder   scanf("%ld",&((*X).lii));
```

(2 Punkte)

Aufgabe 4 (12 Punkte)

a) Stellen Sie die Dezimalzahl -13.3125 als Fließkommazahl im 32 Bit IEEE 754 Standard dar. (3 Punkte)

Vorzeichenbit:

1

Exponent: (mit dem Shift 127)

1	0	0	0	0	0	1	0

Mantisse: (Vergessen Sie nicht die implizite Eins!)

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

b) Stellen Sie in der ersten Zeile die Dezimalzahl -711 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)

-818 :	1	1	1	1	1	1	0	0	1	1	0	0	1	1	1	0
-72 :	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0
$\sum$:	1	1	1	1	1	1	0	0	1	0	0	0	0	1	1	0

c) Welche Ausgaben erzeugt die folgende Code-Sequenz ? Hinweis: `%hd` ist das Ausgabeformat für `short int`. (3 Punkte)

```
1    short int y=50;
2    char str[10]="hallo", *pstr;
3    printf("\n%hd\n",y*y*y);                                  /* 1. Ausgabe: -6072 */
4    for(pstr=str;pstr[0]!=0;++pstr) printf("%c",*pstr+3);     /* 2. Ausgabe: kdoor */
5    printf("\n%d\n",9/4*4/3);                                 /* 3. Ausgabe: 2     */
```

Begründen Sie Ihre Antwort:

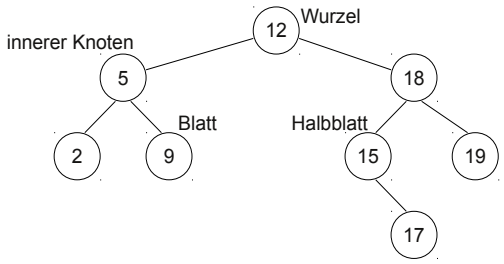
1) $y * y * y \mod 65536 = 125000 \mod 65536 = 59464$
   $59464 = 32768 + 26696 = -32768 + 26696 = -6072$
2) hijk, abcd, lmno, lmno, opqr
3) Auswertung von links nach rechts, ganzzahlige Division:
   $9/4 * 4/3 = (((9/4) * 4)/3) = (2 * 4)/3 = 8/3 = 2$

d) Programmieren Sie eine Funktion `void strcpy(char s1[], char s2[])`, die einen String `s2` in einen String `s1` kopiert. (3 Punkte)

```
1  void strcpy(char s1[], char s2[]){
2    while((*s1++ = *s2++));
3  }
```

Aufgabe 5 (12 Punkte)

Ein binärer Baum ist eine verkettete Liste ohne Kreise, in der jedes Element maximal zwei Nachfolger besitzt, welche "linkes" und "rechtes Kind" genannt werden. Die Elemente des Baumes heißen Knoten. Das erste Element der Liste heißt Wurzel; es besitzt keine Vorgänger. Elemente ohne Kinder heißen Blätter. Elemente mit nur einem Kind heißen Halbblätter. Technisch werden dabei fehlende Kinder durch `NULL`-Zeiger kenntlich gemacht.



a) Definieren Sie eine geeignete Datenstruktur `struct bintree` für die Elemente eines binären Baumes zur Verwaltung von Verkaufsartikeln. Zu jedem Artikel sollen eine ganzzahlige Artikelnummer, die Artikelbezeichnung (maximal 50 Zeichen) und der Preis gespeichert werden.

(3 Punkte)

```
1  struct bintree{
2    int articlenr;
3    char name[51];
4    float price;
5    struct bintree *left, *right;
6  };
```

b) Ein binärer Baum mit der in a) definierten Struktur ist bezüglich der Artikelnummer sortiert, wenn für jeden Knoten die Artikelnummern des linken Teilbaumes kleiner und die des rechten Teilbaumes größer als seine eigene Artikelnummer sind (s. Bild).
Schreiben Sie eine Funktion `float article_price(struct bintree *W, int articlenr)`, die in einem so sortierten binären Baum mit Wurzel `W` nach einem Knoten mit Artikelnummer `articlenr` sucht. Falls der Artikel gefunden wird, soll sein Preis zurückgegeben werden, andernfalls -1.

(4 Punkte)

```
1  float article_price(struct bintree *W, int articlenr){
2    while(W!=NULL && W->articlenr!=articlenr){
3      if(W->articlenr>articlenr){
4        W=W->left;
5        /*printf("L");*/
6      }else{
7        W=W->right;
8        /*printf("R");*/
9      }
10   }
11   if(W==NULL) return -1;
12   else return W->price;
13 }/*article_price*/
```

c) Schreiben Sie eine Funktion

`struct bintree * tree_insert(struct bintree *W, struct bintree *New)`,

die ein neues Element `New` an der richtigen Position eines nach Artikelnummer sortierten binären Baumes mit Wurzel `W` an ein Blatt oder Halbblatt einfügt und die Wurzel des erweiterten Baumes zurück gibt.

(5 Punkte)

```
1   struct bintree * tree_insert(struct bintree *W,struct bintree *New ){
2     struct bintree * E = W;
3     if(W==NULL) return New;
4     while(1){
5       if(E->articlenr==New->articlenr) return W; /* Artikel bereits vorhanden */
6       if(E->articlenr<New->articlenr){
7         if(E->right==NULL){
8           E->right = New;
9           return W;
10        }
11        E = E->right;
12      }else{
13        if(E->left==NULL){
14          E->left = New;
15          return W;
16        }
17        E = E->left;
18      }
19    }
20  }/*tree_insert*/
```