Technische Universität Hamburg-Harburg
Arbeitsbereich Technische Informatik III
Prof. Dr. S.M. Rump

WS 2002

Musterlösung der Klausur zur Vordiplom-Prüfung Programmiermethodik

(Informatik Ingenieurwesen)

12. Februar 2002

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.			

Aufg.	Punkte	Korr.
1		
2		
3		
4		
5		

$\sum$	

Zur Beachtung:
Schreiben Sie bitte weder mit Bleistift noch mit Rotstift.
Kennzeichnen Sie bitte jedes Blatt mit Ihrem Namen und Ihrer Matrikelnummer.
Es sind keinerlei Hilfsmittel in dieser Klausur zugelassen!
Vergessen Sie nicht den „Vorbehalt" zu unterschreiben.

Hinweis zum Ankreuzen:
Stellen Sie bitte sicher, dass Ihre Antwort (Ja/Nein) klar erkennbar ist.

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 (10 Punkte)

Gegeben sei folgende Funktion:

```
 1 int Funk1(char *s, char *t, int n)
 2 {
 3    int loop-var, var= 0, long, b[2];
 4    /*Die ersten Buchstaben der Zeichenkette durchlaufen*/
 5    for( ; var < *n; ++var) {
 6       if(s[0]=='0' || *t==0)
 7          return 0;                          /*Funktion verlassen*/
 8       if(!(*s == *t))
 9          exit;                              /*Schleife verlassen*/
10       ++s; ++t;                             /*Zeiger inkrementieren*/
11    }
12    /*Die Differenz zwischen den aktuellen Buchstaben zurückliefern*/
13    b[2]= *s - *t;
14    if(loop-var >= 0) printf("%d",long);
15    return b[];
16 }
```

Kreuzen Sie bitte Ja für die richtigen Aussagen und Nein für die falschen Aussagen an!

Ja Nein

☒ ☐ Der Funktionsname ist zulässig, weil Großbuchstaben zusammen mit Zahlen erlaubt sind.

☐ ☒ Die Zeile 3 ist nicht korrekt, weil vor jedem Bezeichner ein Datentyp stehen muss.

☒ ☐ Der Bindestrich in dem Bezeichner `loop-var` ist nicht erlaubt.

☐ ☒ Zeile 4 ist inkorrekt, weil dort nur ein Kommentar aber keine Anweisung steht.

☒ ☐ `long` ist kein zulässiger Bezeichner.

☐ ☒ In Zeile 5 muss `+var` stehen, um die Variable um eins zu inkrementieren; `++var` addiert zum Inhalt zwei dazu.

☒ ☐ In Zeile 5 muss `n` stehen und nicht `*n`.

☐ ☒ Die geschweifte Klammer in Zeile 5 muss alleine in einer extra Zeile stehen.

☐ ☒ Die Syntax der `for`–Schleife ist falsch, weil vor dem ersten Semikolon ein Ausdruck stehen muss.

☐ ☒ Die beiden `if` Statements in Zeile 6 & 8 sind falsch, weil mit {} geklammert werden muss.

☐ ☒ Die Schreibweise `s[0]` in der Zeile 6 ist nur für Arrays zulässig, hier aber ist `s` ein Zeiger und somit ist die Schreibweise falsch.

☐ ☒ Der Inhalt des Zeigers `t` ist vom Typ `char` und kann in Zeile 6 nicht mit einer `integer`–Variablen (hier 0) verglichen werden.

☐ ☒ Zeile 8 ist falsch. Sie muss lauten: `if(*s != *t)`

☒ ☐ Der Befehl in Zeile 9 muss `break` lauten, um die Schleife zu verlassen.

☐ ☒ In Zeile 10 stehen zwei Ausdrücke hintereinander, was nicht erlaubt ist.

☐ ☒ In Zeile 11 darf auch `};` stehen.

☒ ☐ Die rechte Seite in Zeile 13 ist erlaubt, weil nur mit dem Inhalt von Zeigern operiert wird.

☒ ☐ In Zeile 13 geht der Index 2 über die Feldgrenze hinaus.

☒ ☐ Zeile 15 ist falsch! Eine richtige Anweisung wäre: `return *b;`

☐ ☒ Für den Inhalt des Zeigers `b` muss mit Hilfe von `malloc()` Speicherplatz reserviert werden.

Aufgabe 2 (17 Punkte)

Schreiben Sie ein lauffähiges Programm, die eine Funktion `binomial` enthält, die für gegebene natürliche Zahlen n und k den Binomialkoeffizienten $\binom{n}{k}$ rekursiv berechnet. Das Ergebnis von `binomial` soll vom Typ `int` sein. Dabei gelten folgende Eigenschaften:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\binom{n}{0} = 1 \qquad \binom{k}{k} = 1$$

Um Zeit zu sparen, sollen bereits berechnete Binomialkoeffizienten in einem Array abgelegt und sofern möglich verwendet werden. Dabei ist der Wert für (n,k) im Array gleich Null, wenn der Binomialkoeffizient noch nicht berechnet wurde, ansonsten ist $\binom{n}{k}$ gespeichert. Das Array soll im Hauptprogramm mit Nullen belegt werden.

```
#include <stdio.h>

double koeffizient[101][51];          /* Globale Variable des Koeffizientenfeldes */

/* Der Binomialkoeffizient wird rekursiv berechnet, wenn die Koeffizienten größer als das Feld sind
oder wenn der Binomialkoeffizient noch nicht im Feld gespeichert ist.
Falls der Wert schon im Feld gespeichert ist, wird dieser zurückgegeben.*/

double binomial(int n, int k) {

   if ((k==0) || (n==k)) return 1;

   if ((n<=100) &&(k<=50)) {
     if (koeffizient[n][k]==0){
       koeffizient[n][k]= binomial((n-1), (k-1)) + binomial((n-1), k);
     }
     return(koeffizient[n][k]);
   }
   else
     return(binomial((n-1), (k-1)) + binomial((n-1), k);
}

/* Im Hauptprogramm wird das globale Feld erstmal mit Nullen initialisiert.
Anschließend werden die Werte von n und k über die Tastertur eingelesen. */

int  main(void) {
  int n,k;
  for(n= 0; n< 101; n++) {
    for(k= 0; k< 51; k++) {
      koeffizient[n][k]= 0;
    }
  }
  while(1){
 printf("Binomialkoeffizienten - Berechnung\n\n");
  printf("Wert von n: ");
  scanf ("%d", &n);
  printf("Wert von k: ");
  scanf ("%d", &k);
  printf("Ergebnis = %g \n", binomial(n,k));}
  return (0);
}
```

Aufgabe 3 (8 Punkte)

Gegeben sei folgende Funktion

```
int vergleich (char *s, char *t, int *n) {
   int k;
   for(k= 0; (*s) && (*s == *t) && (k++ < *n); ++s, ++t);
   *n= k;
   return *s - *t;
}
```

und folgende Zeichenketten:

```
char name[] = "meine_Welt";
char  str[] = "hallo_Welt";
char wort[] = "hallo";
```

Welche Aussagen sind richtig für folgende Funktionsaufrufe im Hauptprogramm?

a) `n= 4;  res = vergleich(str, wort, &n);  printf("n= %d \n", n);`

Ja	Nein	
☒	☐	`res` ist gleich Null
☐	☒	`printf` liefert n= 4

b) `n= 8;  res= vergleich(str, wort, &n);  printf("n= %d \n", n);`

Ja	Nein	
☐	☒	`res` ist gleich Null
☐	☒	`printf` liefert n= 4

c) `n= 4;  res = vergleich(name+6, str+6, &n);  printf("n= %d \n", n);`

Ja	Nein	
☒	☐	`res` ist gleich Null
☒	☐	`printf` liefert n= 4

d) `n= 8;  res = vergleich(name+6, str+6, &n);  printf("n= %d \n", n);`

Ja	Nein	
☒	☐	`res` ist gleich Null
☒	☐	`printf` liefert n= 4

Aufgabe 4 (10 Punkte)

Folgendes Programm sei gegeben:

```
 1 #include<stdio.h>
 2 double dot(int n, double *x, double *y){
 3    double z= 0;
 4    int j;
 5    for(j= 0; j<n; j++)
 6       z+= (*x++) * (*y++)
 7    return z;
 8 }
 9
10 void main(void){
11   double x[10], z[5];
12   int i, n= 4;
13   for(i= 0; i<10 ; i++)
14      { x[i]= i; }
15   for(n= 0; n<5; n++)
16      { z[i]= i; }
17   printf("dot= %d \n", dot(5, x, z));
18   return 0;
19 } /*main*/
```

Welche Aussagen sind richtig?

Ja Nein

- ☐ ☒ Zeile 3: Die Deklaration und die Initialisierung müssen in zwei getrennten Anweisungen stehen.
- ☐ ☒ Zeile 4 ist inkorrekt, da im Unterprogramm keine Variablen deklariert werden dürfen.
- ☒ ☐ In Zeile 7 darf auch `return(z);` stehen.
- ☐ ☒ Zeile 13: es muss `i<=10` heißen, weil man sonst nicht alle Felder von x belegt.
- ☒ ☐ Zeile 14 und 16 kann man die geschweiften Klammern {} weglassen.
- ☐ ☒ Zeile 15 ist falsch, da n bereits mit 4 initialisiert wurde.
- ☐ ☒ Das Hauptprogramm ist falsch, weil die Dimension der Arrays x und z ungleich sind ($10 \neq 5$).
- ☒ ☐ Im Hauptprogramm wäre der Aufruf `dot(5, x+5, z)` zulässig.
- ☐ ☒ In Zeile 17 kann man die Anführungszeichen um %d gegen Hochkommata ersetzen.
- ☐ ☒ In Zeile 17 wird mit \n die Variable n falsch verwendet.
- ☒ ☐ %d ist falsch, es müsste %f heißen.
- ☒ ☐ Der Aufruf `dot(n, z, x)` in Zeile 17 wäre korrekt.
- ☒ ☐ Der Aufruf `dot(5, x[5], z[5])` wäre korrekt.
- ☐ ☒ Der Aufruf der Funktion dot ist falsch, da die Parameter x und y heißen müssen anstatt x und z.
- ☒ ☐ Zeile 18 ist falsch, weil man keinen Wert zurückgeben darf.
- ☐ ☒ Die Variable z ist unzulässig, da sie im Haupt- und Unterprogramm benutzt wird.
- ☐ ☒ Anstatt `z+= (*x++)*(*y++)` kann man schreiben `z+= *(x*y); x++; y++;`
- ☒ ☐ Anstatt `z+= (*x++)*(*y++)` kann man schreiben `z+= (*x)*(*y); x+= 1; y+= 1;`
- ☒ ☐ Anstatt `z+= (*x++)*(*y++)` kann man schreiben `z+= x[0]*y[0]; ++x; ++y;`
- ☐ ☒ Anstatt `z+= (*x++)*(*y++)` kann man schreiben `z= z + ((*x)*(*y))++;`

Aufgabe 5 (15 Punkte)

Schreiben Sie ein lauffähiges Programm zur Bestimmung aller Primzahlen zwischen 2 und 1000. Legen Sie dazu ein Feld p der Länge 1000 an. Dabei soll p[i] angeben, ob die Zahl i eine Primzahl ist. Die Vorgehensweise zur Berechnung der Primzahlen ist die folgende:
Zunächst werden alle Elemente des Feldes p als Primzahl initialisiert. Für alle Zahlen i von 2 bis 1000 wird dann folgendes durchgeführt:

a) Ist p[i] noch als Primzahl gesetzt, so ist i eine Primzahl und i wird ausgegeben.

b) Alle Vielfachen von i können keine Primzahlen sein und werden daher gestrichen (d.h alle p[j], $j = 2i, 3i...$ sind keine Primzahlen).

Beachten Sie, dass die Zahl 1 per definitionem keine Primzahl ist.

```
/* Deklaration eines Feldes der Größe 1000 und Initialisierung mit 1.
   Das heißt alle Zahlen sind erstmal als Primzahlen deklariert.*/

int main(void){
  int p[1000];
  int n; int k;
  for(n= 0; n< 1000; n++){
    p[n]=1;
  }

/* Für alles p[n] die auf 1 sind wird der Wert ausgegeben und alle vielfachen von p[n] werden
   auf 0 gesetzt.
   Bsp.:  p[n]=3 ist Primzahl das heißt alle Vielfachen wie 6,9,12... können keine Primzahlen
   mehr sein und werden auf 0 gesetzt.*/

  for(n= 2; n< 1000; n++){
    if(p[n]==1){
      printf("%d \n",n);
      for(k= n; k< 1000; k+= n)
        p[k]=0;
    }
  }
  return 0;
}
```