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

WS 2007/2008

Musterlösung der Klausur zur Zwischen-Prüfung Programmiermethodik

(Informatik-Ingenieurwesen / Informationstechnologie)

9. Januar 2008

Sie haben 40 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.		

Es sind alle Aufgaben zu bearbeiten. Insgesamt können bis zu 28 Punkte erreicht werden.

Aufg.	Punkte	Korr.
1		
2		
3		

$\sum$	

Zur Beachtung:
Schreiben Sie bitte weder mit Bleistift noch mit Rotstift.
Es sind keinerlei Hilfsmittel in dieser Klausur zugelassen!
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 (8 Punkte)

Folgendes Programm soll den Wochentag anhang eines durch den Benutzer eingegebenen Datums nach der Formel von C. F. Gauß berechnen und ausgeben. Leider sind dem Programmierer einige Fehler unterlaufen. Den Fehlermeldungen des Compilers kann er entnehmen, dass genau 8 Fehler vorhanden sind.

Helfen Sie dem Programmierer diese Fehler zu finden. (**Falsch erkannte Fehler werden negativ bewertet!**)

```
1  #include <stdio.h>
2
3  int wochentag(int d, int m, int year){
4      int c=0,y=0,w=0,z=0;
5      int julianMonate={11,12,1,2,3,4,5,6,7,8,9,10};
6      m=julianMonate[m-1];
7      if (m==11 || m==12) year--;
8      y=year%100;
9      c=year/100;
10     z=d+(int)(2.6*m-0.2)+y+y/4+c/4-2*c;
11     w=z%7;
12     if (w<0) w+=7;
13     return w
14 }
15
16 int main(void)
17     int d=0,m=0,year=0,w=0;
18     char monate[]={"Januar","Februar","Maerz","April","Mai","Juni","Juli",
19                    "August","September","Oktober","November","Dezember"};
20     char wochentage[]={"Sonntag","Montag","Dienstag","Mittwoch",
21                        "Donnerstag","Freitag","Samstag");
22     printf("\nBitte geben Sie das Datum ein:\n");
23     printf("Tag :");   scanf("%d",&d);
24     printf("Monat: "); scanf("%d",*m);
25     printf("Jahr: ");  scanf("%d",year);
26     w=wochentag(d,m,year);
27     printf("\n%d %s %d ist ein %s\n\n",d,monate[m-1],year,wochentage[w]);
28     return 0;
29 }
```

Lösung:

Der angegebene Programmcode enthält insgesamt 8 Fehler. Für jeden korrigierten Fehler wird ein Punkt vergeben. Für falsch erkannte Fehler wird jeweils ein Punkt abgezogen. (Negative Punkte übertragen sich nicht auf andere Aufgaben)

Nachfolgend ist der korrekte Code zu sehen. Die Anzahl der rechts angegebenen Punkte entspricht dabei der Anzahl vorhandener Fehler in der jeweiligen Zeile.

```
1   #include <stdio.h>
2
3   int wochentag(int d, int m, int year){
4       int c=0,y=0,w=0,z=0;
5       int julianMonate[]={11,12,1,2,3,4,5,6,7,8,9,10};                          /*(1 Punkt)*/
6       m=julianMonate[m-1];
7       if (m==11 || m==12) year--;
8       y=year%100;
9       c=year/100;
10      z=d+(int)(2.6*m-0.2)+y+y/4+c/4-2*c;
11      w=z%7;
12      if (w<0) w+=7;
13      return w;                                                                  /*(1 Punkt)*/
14  }
15
16  int main(void){                                                                /*(1 Punkt)*/
17      int d=0,m=0,year=0,w=0;
18      char *monate[]={"Januar","Februar","Maerz","April","Mai","Juni","Juli", /*(1 Punkt)*/
19                      "August","September","Oktober","November","Dezember"};
20      char *wochentage[]={"Sonntag","Montag","Dienstag","Mittwoch",           /*(1 Punkt)*/
21                          "Donnerstag","Freitag","Samstag"};                  /*(1 Punkt)*/
22      printf("\nBitte geben Sie das Datum ein:\n");
23      printf("Tag :");   scanf("%d",&d);
24      printf("Monat: "); scanf("%d",&m);                                         /*(1 Punkt)*/
25      printf("Jahr: ");  scanf("%d",&year);                                      /*(1 Punkt)*/
26      w=wochentag(d,m,year);
27      printf("\n%d %s %d ist ein %s\n\n",d,monate[m-1],year,wochentage[w]);
28  return 0;
29  }
```

Kommentar:

In der dritten Zeile wurden bei der Deklaration der array-Variablen `julianMonate` die eckigen Klammern `[]` vergessen. In der Zeile 13 ist die Anweisung (wie immer) mit einem Semikolon abzuschließen. In den Zeilen 18 und 20 muessten die beiden Variablen `monate` bzw. `wochentage` als Arrays vom Typ Zeiger auf Character (`char*`) deklariert werden. In Zeile 21 ist die runde Klammer falsch und ist durch eine geschweifte Klammer zu ersetzen. In den Zeilen 24 und 25 wird die Bibliotheks-Funktion `scanf` mit falschen Parametertypen aufgerufen.

Aufgabe 2 (8 Punkte)

Nehmen Sie bei dieser Aufgabe an, dass vorzeichenbehaftete Datentypen im 2-er Komplement dargestellt werden.

a) Stellen Sie die Zahlen -5 und 15 als `short int` (binär) dar.

Lösung:

-5 :	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1

(3 Punkte)

15 :	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(1 Punkt)

b) Welche Ausgabe erzeugt folgende Code-Sequenz:

```
101    ...
102    short int k=-32768;
103    k=-k;
104    printf("%d \n",k);
105    k=32767;
106    k=k+1;
107    printf("%d \n",k);
108    ...
```

Ausgabe:

-32768 (1 Punkt)

-32768 (1 Punkt)

Begründen Sie Ihre Antwort:

Begründung:

Der erste Fall läßt sich folgendermaßen erklären: Das 2-er Komplement einer Zahl wird durch die Inversion aller Bits und anschlißende Addition einer 1 gebildet. Für `k=-32768` gilt die binäre Darstellung: 1000000000000000. Die Inversion aller Bits liefert 0111111111111111 und eine Addition der 1 führt schließlich zu demselben Bitmuster. Folglich ergibt sich: `-k==-32768`. (1 Punkt)
Im zweiten Fall stellt `k` die größte positive Zahl für diesen Datentyp dar. Die Addition von 1 bewirkt einen Überlauf, was bei der Darstellung im 2-er Komplement zur betragsgrößten Zahl führt. (An dieser Stelle ist es wichtig zu erwähnen, dass dieses Ergebnis auch von der Implementierung des Compilers abhängig ist.) (1 Punkt)

Aufgabe 3 (12 Punkte)

Schreiben Sie ein Programm, welches alle Primzahlen zwischen 2 und 100 bestimmt und auf dem Bildschirm ausgibt. Gehen Sie dafür folgendermaßen vor: Legen Sie zuerst ein Feld `prim` der Länge 100 an und initialisieren Sie alle seine Elemente mit 1. Dabei soll `prim[i]` angeben, ob die Zahl `i` eine Primzahl ist. Zur Bestimmung der Primzahlen wird für alle `i` von 2 bis 100 geprüft, ob `i` noch als Primzahl anzusehen ist, d.h. `prim[i]==1` oder nicht. Wenn `i` eine Primzahl ist, dann werden alle Vielfachen davon als keine Primzahl markiert. Dafür werden für alle `j=2*i,3*i...` die array-Elemente `prim[j]` gleich 0 gesetzt. Bitte beachten Sie, dass 1 keine Primzahl ist und setzen Sie schon zu Beginn des Algorithmus entsprechend die Variablen.

Lösung:

Die genaue Punktevergabe entnehmen Sie bitte der unten stehenden Tabelle. Ein Beispiel zur Punktevergabe finden Sie im Programm-Code der Musterlösung rechts als Kommentar angegeben.

```
1   #include <stdio.h>                                                    /*(1 Punkt )*/
2   #define LEN 100
3
4   void eratosthenes(int* prim){
5       int i=0,j=0;
6       /* 0 wird nicht weiter betrachtet*/
7       /* 1 wird als keine Primzahl markiert*/
8       prim[1]=0;                                                       /*(1 Punkt )*/
9       /* angefangen mit der 2 werden alle Vielfachen
10         der "aktuellen Primzahl" gleich 0 gesetzt */
11      for(i=2;i<LEN;++i){                                              /*(6 Punkte)*/
12          if (prim[i]){
13              for(j=2*i;j<LEN;j=j+i){
14                  prim[j]=0;
15              }
16          }
17      }
18  }
19
20  int main(void){                                                      /*(1 Punkt )*/
21      int i=0,prim[LEN];
22      /* initialisiere das Array mit Zahlen von 1-100 */
23      for(i=0;i<LEN;i++) prim[i]=1;                                    /*(1 Punkt )*/
24      /* Bestimmung der Primzahlen nach Eratosthenes */
25      eratosthenes(prim);
26      /* Ausgabe der Primzahlen */
27      for(i=1;i<LEN;i++){                                              /*(2 Punkte)*/
28          if (prim[i])
29              printf("%d ",i);
30      }
31      printf("\n");
32      return 0;
33  }
```

umgesetzte Programmabschnitte	**Punkte**
Präprozessor-Direktiven	1
main-Signatur	1
Deklaration lokaler Variablen	1
korrekte Ausgabe	2
1 als nicht Primzahl markiert	1
richtige Umsetzung des beschr. Algor.	6