KLAUSUR 3 SAE E1FI 03.04.2003 Zeit: 60 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Die Lösungen müssen als ablauffähige C-Programme dargestellt werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenoommen werden.
→ Die rote Farbe darf nicht benutzt werden.
→ Aufgabenblätter bitte auch abgeben.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Die Programme müssen ausgedruckt werden.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Geben Sie die syntaktischen Fehler und die Fehler, die zur Laufzeit entstehen können, an.

```
#include "stdafx.h"
#include <stdio.h>

void main()
{
int i = 3;
int v[3]={10,9,15,8};
double w[i];

v[2] = 10;
w[i+1]=v[2];
printf("r=%d\n",v[i-3]);
}
```

2) Schreiben Sie ein C-Programm, das die Umsätze (grösser Null) einer Firma in ein Feld einliest.
Die Umsätze müssen über Tastatur eingegeben werden. Wenn für einen Umsatz ein Wert kleiner als Null eingegeben wird, wird die Eingabe beendet. Dann wird der Gesamtumsatz (die Summe dieser Umsätze) berechnet und auf dem Bildschirm ausgegeben.

Bemerkung:
1) EVA-Prinzip beachten
2) Es dürfen keine nicht reservierten Zellen eines Feldes überschrieben werden.

KLAUSUR 4 SAE E1FI 3.07.2003 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
alle, ausser programmierbaren Geräten

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Die Lösungen müssen als ablauffähige C-Programme dargestellt werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenoommen werden.
→ Die rote Farbe darf nicht benutzt werden.
→ Aufgabenblätter bitte auch abgeben.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) In einem C-Programm wurde folgende Variable deklariert:

```
char v[4];
```

Geben Sie Ihren Kommentar zu folgenden Anweisungen ab:

```
v[0] = 'v';
v[4] = 'r';
v[-1] = 'r';
```

2) Schreiben Sie ein C-Programm, das jede Zahl in einem Feld um 1 erhöht.

3) Schreiben Sie ein C-Programm, das ein Feld in umgekehrter Reihenfolge in ein anderes Feld kopiert.
Beispiel:
aus

4	7	1	3	2

wird:

2	3	1	7	4

4) Schreiben Sie eine Funktion, die das Volumen eines Quaders berechnet.
Rufen Sie die Funktion im Hautpprogramm auf.

KLAUSUR 4 SAE E1FI1 3.07.2003 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
alle, ausser programmierbaren Geräten

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Die Lösungen müssen als ablauffähige C-Programme dargestellt werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenoommen werden.
→ Die rote Farbe darf nicht benutzt werden.
→ Aufgabenblätter bitte auch abgeben.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) a) Welche Fehlermeldungen liefert der Compiler ?
b) Welchen Wert hat erg nach jeder Anweisung (falls diese syntaktisch korrekt ist) ?

```
#include "stdafx.h"
#include <stdio.h>

void main()
{
double f(int i, int j);
double erg = 12;
int m = 4;
int n = 5;

erg = f(i, j);          // erg =
erg = f(2, m);          // erg =
erg = f(m, n);          // erg =
erg = f(2, 4, 6);       // erg =
erg = f(n, m);          // erg =
erg = f(m, 4);          // erg =
erg = f(6, 4);          // erg =
f(12, 4);
}

double f(int i; int j)
{
double erg;
erg = i + j;
return(erg);
}
```

2) Schreiben Sie eine Funktion, die das Volumen eines Quaders berechnet.
Rufen Sie die Funktion im Hautpprogramm auf.

3) Schreiben Sie eine Funktion, die den Vorgänger (die um 1 kleinere Zahl) einer Zahl liefert.
Rufen Sie die Funktion im Hautpprogramm auf.

KLAUSUR 1 SAE E1FI 25.11.2003 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):
- → Alle Aufgaben müssen bearbeitet werden.
- → Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- → Aufgabenblätter bitte auch abgeben.
- → Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- → Die rote Farbe darf nicht benutzt werden.
- → Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- → Selbsterklärende Variablennamen benutzen.
- → Programme müssen benutzerfreundlich sein.
- → EVA-Prinzip muss benutzt werden.
- → Einrücken der entsprechenden Programmteile.
- → Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Welche Eigenschaften hat ein Algorithmus ?

2) Warum ist folgende Anweisung nach Verwendung der scanf-Funktion sinnvoll ?
```
fflush(stdin);
```

3) Warum gibt es bei der Ausführung folgender, syntaktisch korrekter Anweisung Probleme ?
```
int d;
d = 6/(1/3);
```

4) Ist die Verwendung des cast-Opeartors (double) in der folgenden 2 syntaktisch korrekten Anweisungen unbedingt nötig ? Begründen Sie !
```
double d, r;
int i;
d = (double)2 * 3.14 * r;
i = (int)(4 * 2.718* i);
```

5) Erzeugt der Compiler in der folgenden, syntaktisch korrekten Anweisung eine Warnung ? Begründen Sie !
```
float d;
d = 2 * 3.14 * d;
```

6) Welche Wert(e) können die folgenden syntaktisch korrekten Ausdrücke jeweils annehmen ?
```
z = 10
z == 10
```

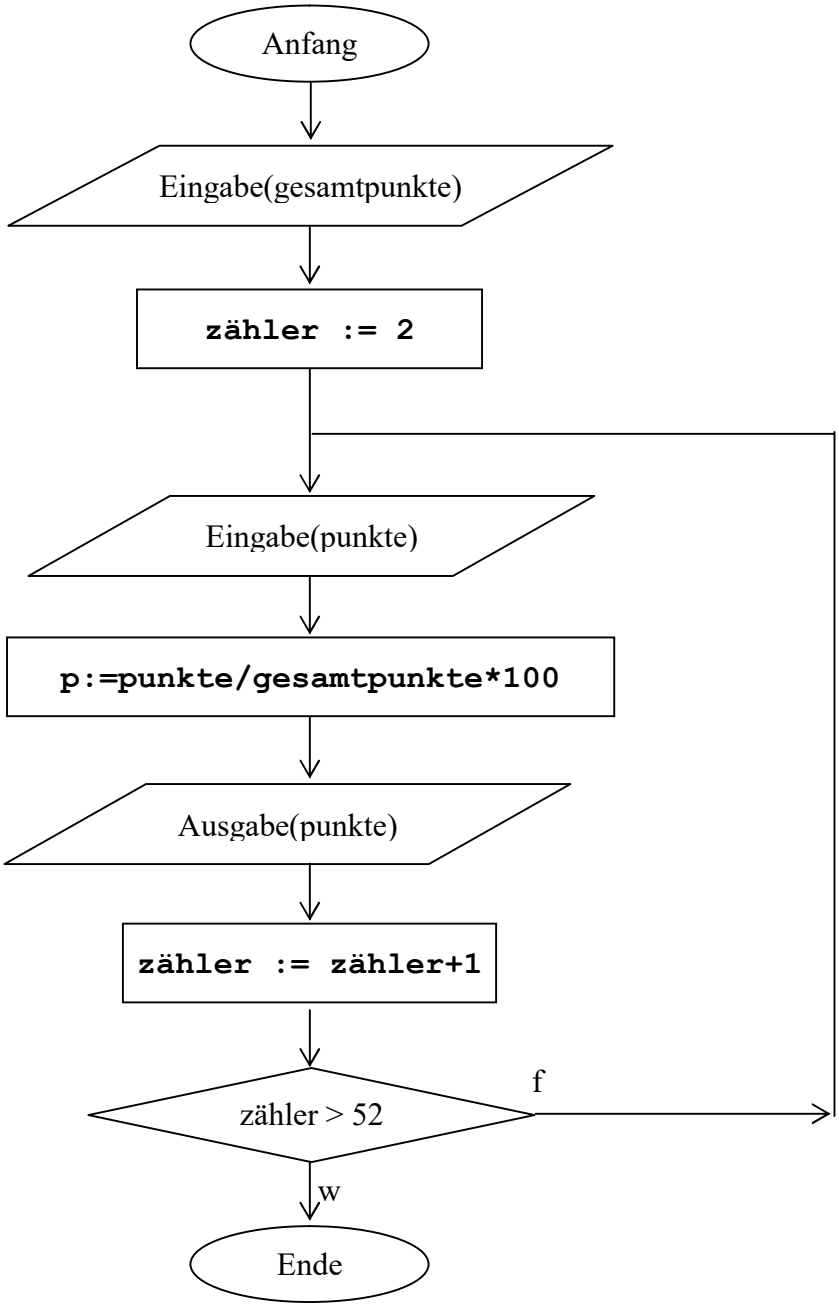
7) Geben Sie ein Beispiel an, wann die folgenden syntaktisch korrekten Ausdrücke den gleichen Wert haben und ein Beispiel, wann sie nicht den gleichen Wert haben.
```
(a==b)==c
(a==b)&&(b==c)
```

8) Geben Sie ein Beispiel an, wann die folgenden syntaktisch korrekten Ausdrücke den gleichen Wert haben und ein Beispiel, wann sie nicht den gleichen Wert haben.
```
(x==2)||4||6
(x==2)||(x==4)||(x==6)
```

9)
Es soll von 50 Schülern jeweils der Prozentsatz der erreichten Punkte bzgl. der Gesamtpunktzahl in einer Klassenarbeit errechnet und ausgegeben werden.

1) Ist die folgende Lösung korrekt ? Begründen Sie !
Falls sie nicht korrekt ist, verbessern sie die Lösung.
Suchen Sie dazu eine geeignete Stelle im Programm aus, an der Sie das Programm gedanklich bei jedem Schleifendurchgang anhalten und dort den Wert der Variablen zähler und die Anzahl der ausgegebenen Schüler protokollieren:

Anzahl der ausgegebenen Schüler	Wert der Variablen zähler



Lösungen:
1) Ein Algorithmus ist ein Verfahren zur Lösung eines gegebenen Problems. Er ist eindeutig, endlich und schrittweise.

2) Mit fflush(stdin) werden evtl.noch im Tastaturpuffer stehende Zeichen gelöscht.

3)
```
d = 6/(1/3);
```
1/3 hat den Wert 0, also wird durch Null dividiert.
Diese Anweisung wird von keinem Mikroprozessor ausgeführt.

4) a)
```
d = (double)2 * 3.14 * r;
```
2 ist ein Integer-Wert. Bei Multiplikation mit einem double-Wert (hier 3.14) wird der Integer-Wert 2 vor der Multiplikation implizit (automatisch, ohne Zutun des Programmierers) in einen double-Wert umgewandelt. Das bedeutet, daß die cast Umwandlung mit (double) nicht gemacht werden muß.

b)
```
i = (int)(4 * 2.718* i)
```
Der Wert von 4 * 2.718* i ergibt einen double-Wert. Wenn man die explizite Umwandlung mit dem cast Operator (int) wegläßt, wird dieser double-Wert in der Integer-Wert Variablen i abgespeichert. Dadurch entsteht ein Datenverlust. Dies meldet der Compiler.

5)
```
d = 2 * 3.14 * d;
```
Der Wert von 2 * 3.14 * d ergibt einen double-Wert.
Dieser double-Wert wird in der float-Variablen d abgespeichert. Dadurch kann ein Datenverlust entstehen. Dies meldet der Compiler.

6)
Der Ausdruck z = 10 hat immer den Wert 10.
Der Ausdruck z ==10 hat den Wert 0 oder 1. Also sind die Werte nie gleich, sondern immer verschieden. Die Frage beinhaltet also einen Fehler.

7)
a) a = 1, b= 2, c = 3
```
(1==2)==3          (1==2)&&(2==3)
   0 == 3            0  ==  0
      0                  0
```

b) a = 1, b= 2, c = 0
```
(1==2)==0          (1==2)&&(2==0)
   0 == 0            0  ==  0
      1                  0
```
8)
a) x=2
```
(2==2)||4||6       (2==2)||(2==4)||(2==6)
   1  ||4||6          1  ||   0  ||   0
      1                     1
```

b) x = 3
```
                   (3==2)||(3==4)||(3==6)
                      0   ||   0  ||   0
                               0
```

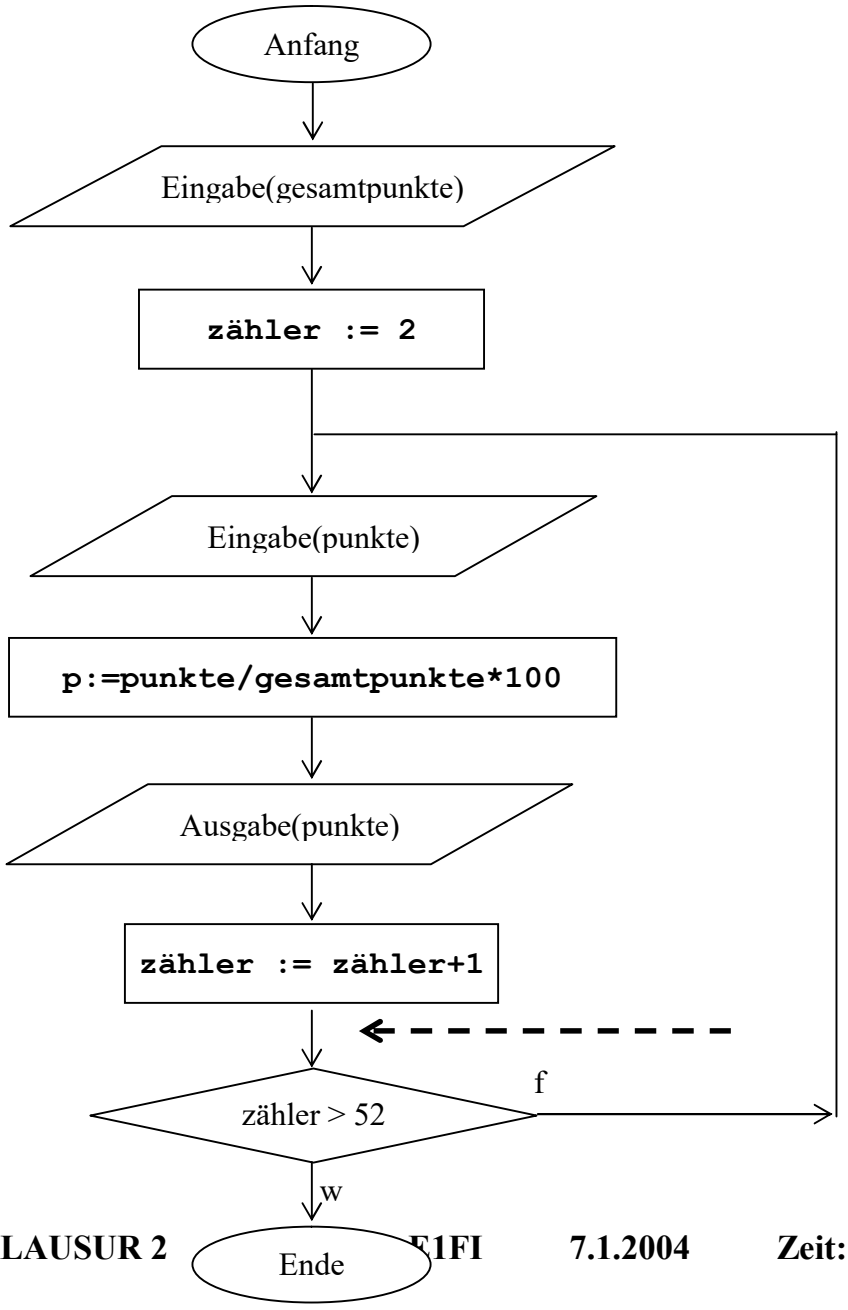
```
(3==2)||4||6
   0  ||4||6
      1
```

9)

Anzahl der ausgegebenen Schüler	Wert der Variablen zähler
1	3
...	...
2	4
51	53

a) Es werden nicht 50, sondern 51 Schüler ausgegeben.
Begründung: siehe Tabelle.

b) Korrektur:
Statt: zähler > 52
besser: zähler >= 52



Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):
→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Der folgende, syntaktisch korrekte Programmausschnitt soll feststellen, ob eine über Tastatur eingegebene ganze Zahl Element der folgenden Menge ist: {2; 4; 6; 8}:
a) Geben Sie einen **konkreten** Wert für z an, für den das Programm eine falsche Ausgabe erzeugt. Begründen Sie.
Bemerkung: Es wird vorausgesetzt, dass der Benutzer auch eine ganze Zahl eingibt.

```
Programmausschnitt:
...
if(z==(2||4||6||8))
  printf("Die Zahl ist entweder 2, 4, 6, oder 8");
else
  printf("Die Zahl ist nicht 2, 4, 6, oder 8");
...
```

b) Ändern Sie das Programm so ab, dass es semantisch korrekt wird (das oben Angegebene macht).

2) Erstellen Sie von folgender IF...ELSE...IF-Kette (Programmteil) ein Struktogramm:

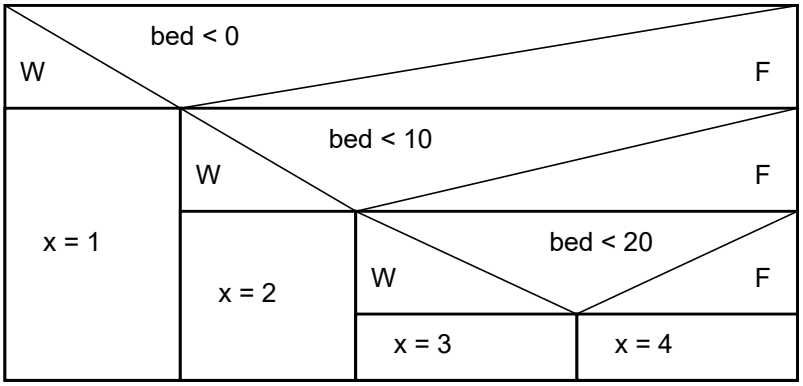
```
...
if(bed<0){
  x = 1;
}
else if(bed < 10){
  x = 2;
}
else if(bed < 20){
  x = 3;
}
else{
  x = 4;
}
```

3) Erstellen Sie ein C- Programm, das das Volumen und die Oberfläche eines Quaders mit den Seiten a, b und c berechnet.
Die Seitenlängen (Einheit: cm) müssen über Tastatur eingegeben werden.
Wenn für eine Seitenlänge ein Wert kleiner oder gleich 0 eingegeben wird, kann der Benutzer keine weitere Seitenlänge mehr eingeben und außerdem wird dann auch kein Volumen bzw. Oberfläche mehr berechnet, sondern eine entsprechende Meldung ausgegeben.
Nur wenn alle 3 Seitenlängen grösser Null sind, soll das Volumen und die Oberfläche berechnet werden.

Lösung:

1)
Für z = 1 ist der Ausdruck `z==(2||4||6||8)` wahr und hat dann den Wert 1.
Es wird dann ausgegeben: "Die Zahl ist entweder 2, 4, 6, oder 8"

2)



3)

```
void main(){
      double l,b,h,flaeche,volumen;
      int weiter;
      printf("Laenge eingeben\n");
      scanf("%lf",&l);
      if(l>=0)
            weiter=1;
      else
            weiter=0;

      if(weiter==1){
            printf("Breite eingeben\n");
            scanf("%lf",&b);
            if(b>=0)
                  weiter=1;
            else
                  weiter=0;
      }
      if(weiter==1){
            printf("Hoehe eingeben\n");
            scanf("%lf",&h);
            if(h>=0)
                  weiter=1;
            else
                  weiter=0;
      }
      if(weiter==1){
            volumen = l * b * h;
            flaeche = 2*(l*b+l*h+b*h);
      }

      if(weiter==1){
            printf("Volumen=%f\n",volumen);
            printf("Flaeche=%f\n",flaeche);
      }
      else{
            printf("Seitenlaenge kann nicht negativ sein\n");
      }
}
```

E

V

A

KLAUSUR 3 SAE E1FI 24.3.2004 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) In einem C-Programm wurde folgende Variable deklariert:

```
char v[4];
```

Geben Sie Ihren Kommentar zu folgenden Anweisungen ab:

```
v[0] = 'v';
v[4] = 'r';
v[-1] = 'r';
```

2) Wieviel Speicherplatz (in Bytes) benötigt der durch die folgende Deklaration reservierte Speicher ? Begründen Sie !

```
double werte[10][20];
```

3) Ein Programm fordert einen Anwender auf, in 100 verschiedenen Eingabefenstern Angaben (z.B. Name, Vorname, Wohnort, usw.) zu seiner Person (in Form von Zeichenketten)zu machen.
Der dafür reservierte Speicher soll vom Programmierer am Programmanfang mit '\0' initialisiert werden. Sollen für das Programm 100 eindimensionale oder ein zweidimensionales Feld verwendet werden ?
Beurteilen und begründen Sie dies aus der Sicht des Programmierers, der möglichst wenig Programmcode schreiben will.

4) Es wird das folgende zweidimensionale Feld deklariert:

```
char v[3][5];
```

Was ist v [1] ? Exakte Beschreibung !

5) Erstellen Sie ein Struktogramm (**kein** Programm), welches Zahlen wie folgt kodiert:

a) Lesen Sie Zahlen über Tastatur in das ganze Feld felda

b) Geben Sie die Zahlen von felda auf dem Bildschirm (beginnend mit der nullten Zelle des Feldes) aus.

c) Kopieren Sie die Zahlen von felda in das Feld feldb (beginnend mit der nullten Zelle des Feldes)

d) Die Zahlen in feldb sollen in umgekehrter Reifenfolge in felda kopiert werden.
Beispiel:

felda (vorher) :	5	1	6	3	5
felda (nachher) :	5	3	6	1	5

e) Geben Sie die Zahlen von felda auf dem Bildschirm (beginnend mit der nullten Zelle des Feldes) aus.

Bemerkungen:

1) Die Feldlänge soll vom Programmierer ohne aufwändige Bemühungen an genau einer Stelle leicht zu verändern sein.
(und muß dann neu compiliert werden) und soll zuerst einmal den Wert 5 haben.

Lösungen:

1)
```
v[0] = 'v';      // richtig
v[4] = 'r';      // falsch: nicht reservierter Speicher
v[-1] = 'r';     // falsch: das erste Feld beginnt bei 0
```

2)
10 * 20 * sizeof(double) = 1600 Byte

3)
Bei 100 eindimensionalen Feldern braucht man 100 Schleifen, dagegen ist bei einem zweidimensionalen Feld nur eine Schleife nötig.

4)
v [1] ist ein eindimensionales Feld, das aus den folgenden Elementen besteht:
v[1][0], v[1][1], v[1][2], v[1][3], v[1][4]

KLAUSUR 3 SAE E1FI Nachtermin1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Ein zweidimensionalen Feld
int zv1[3]3]
ist mit bestimmten Zahlen vorbelegt und soll vor seiner Versendung über eine Datenleitung aus Sicherheitsgründen verschlüsselt werden. Dies geschieht dadurch, daß die 1. Zeile in die 1. Spalte, die 2. Zeile in die 2. Spalte und die 3. Zeile in die 3. Spalte des folgenden zweidimensionalen Feldes kopiert wird:
int zv1[3]3]

Beispiel

1	2	3
4	5	6
7	8	9

--->

1	4	7
2	5	8
3	6	9

Schreiben Sie ein C-Programm, das dies (für eine beliebige Zahlenbelegung, nicht nur für die des obigen Beispiels) realisiert.

2) Schreiben Sie ein C-Programm, in dem der Anwender eine Zeichenkette über Tastatur eingibt und in der ersten Zeile eines zweidimensionalen Feldes abspeichert.
Dann soll das Programm diese eingegebene Zeichenkette in umgekehrter Reihenfolge in der zweiten Zeile des zweidimensionalen Feldes abspeichern und danach auf dem Bildschirm ausgeben.

Beispiel:
Eingabe: E1FI
Ausgabe: IF1E

Bemerkungen:
1) Die Feldlänge soll vom Programmierer ohne aufwändige Bemühungen an genau einer Stelle leicht zu verändern sein.
(und muß dann neu compiliert werden) und soll zuerst einmal den Wert 5 haben.

KLAUSUR 4 SAE E1FI 23.6.2004 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Geben Sie die **syntaktischen** Fehler in den folgenden Funktionen an.
Welche Werte haben die Variablen ii, ff, dd, usw. nach dem jeweiligen Aufruf im Hauptprogramm main ? Geben Sie diese Werte jeweils im mit // bezeichnet Kommentar an.

```
void g1(int i){
  return(i);
}

void g2(float *f){
  f = *f;
}

void g3(double *d){
  *d = *d * *d;
}

void h1(int i){
  i = 2 * i;
}

void main(){
  int ii = 5;
  float ff = 4;
  float erg = 3;
  double dd = 2;

  ii = g1(2);       // ii  =
  g2(&ff);          // ff  =
  g3(&dd);          // dd  =
  h1(ii);           // ii  =
}
```

2) Warum kann es bei den folgenden Programmierzeilen zu einem Programmabsturz kommen ?

```
int *i
*i = 123;
```

3)
Schreiben Sie ein Funktion, die den Benzinverbrauch pro 100 km berechnet.
Erzeugen Sie dazu auch einen Aufruf im Hauptprogramm.

4) Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion:

```
/*************************************************************/
/**                                                         **/
/**  int benzin(double km, double l, double lp,             **/
/**             double *l100, double *e100)                 **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) double km:    Strecke (in Km)
  (i) double l:     Verbrauch (in Liter)
  (i) double lp:    Literpreis Benzin
  (o) double l100:  Benzinverbrauch in Liter pro 100 Km
  (o) double e100:  Benzinpreis in Euro pro 100 Km

Return:
  -1: km <= 0
   0: sonst

Beschreibung:
  Berechnet in Abhängigkeit von den gefahrenen Kilometern
  (km), den dafür benötigten Litern Benzin (l) und dem
  Benzinliterpreis (lp) den Literverbrauch Benzin pro 100 Km
  (l100) und die Kosten in Euro für das Benzin pro 100 Km
  (e100).
  Falls eine Kilometerzahl <= 0 angegeben wird, wird -1
  zurückgegeben, ansonsten 0.
*/
```

KLAUSUR 4 SAE E1FI Nachtermin1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) a) Schreiben Sie ein Funktion, die (in Abhängikeit eines weiteren Parameters) aus einem Bruttopreis und der Mehrwertsteuer den Nettopreis, bzw. aus das einem Nettopreis (und der Mehrwertsteuer) den Bruttopreis berechnet.

b) Erzeugen Sie dazu auch einen Aufruf im Hauptprogramm.

2) a) Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion:

```
/*************************************************************/
/**                                                         **/
/**  int widerstand (double *R1, double *R2)                **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i/o) double R1:    erste Widerstand (in Ohm) bzw.
                      Gesamtwiderstand bei Serienschaltung
  (i/o) double R2:    zweite Widerstand (in Ohm) bzw.
                      Gesamtwiderstand bei Parallelschaltung

Return:
  -1: R1 < 0 oder R2 < 0
   0: sonst

Beschreibung:
  Berechnet den parallelen und seriellen Gesamtwiderstand der
  2 Widerstände R1 und R2.
  Der Gesamtwiderstand bei der Serienschaltung wird in *R1
  zurückgegeben, der Gesamtwiderstand bei der
  Parallelschaltung wird in *R2 zurückgegeben.
  Falls ein Widerstand < 0 ist, wird -1 zurückgegeben,
  ansonsten 0.
*/
```

b) Erzeugen Sie dazu auch einen Aufruf im Hauptprogramm.

KLAUSUR 1 SAE E1FI 17.11.2005 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Welche Eigenschaften hat ein Algorithmus ? (3 P)

2) Was ist die ANSI Norm ? (2 P)

3) Was ist ein Literal ? (2 P)

4) Kann die Ganzzahl 29999 in einer zwei Bytes grossen Integer-Zahl abgespeichert werden ? Begründen Sie! (4 P)

5) Warum ist folgende Anweisung nach Verwendung der scanf-Funktion sinnvoll ?

```
fflush(stdin);
```

(3 P)

6) Schreiben Sie ein C-Programm, das die Summe, die Differenz, das Produkt und den Quotienten zweier Zahlen berechnet:
Der Anwender muss zwei Zahlen (1. Zahl Datentyp float, zweite Zahl Datentyp double) eingeben.
Dann muss die Summe, die Differenz, das Produkt und der Quotient berechnet werden.
Die Ergebnisse müssen jeweils als **ganze** Zahl abgespeichert (Nachkommateil abschneiden) werden. EVA-Prinzip verwenden ! (20 P)

7) Warum gibt es bei der Ausführung folgender, syntaktisch korrekter Anweisung Probleme ?

```
int d;
d = 6/(1/3);
```

(4 P)

8) Ist die Verwendung des cast-Opeartors (double) in der folgenden 2 syntaktisch korrekten Anweisungen unbedingt nötig ? Begründen Sie !

```
double d, r;
int i;
d = (double)2 * 3.14 * r;
i = (int)(4 * 2.718* i);
```

(4 P)

9) Erzeugt der Compiler in der folgenden, syntaktisch korrekten Anweisung eine Warnung ? Begründen Sie !

```
float d;
d = 2 * 3.14 * d;
```

(4 P)

Lösungen:

1) Ein Algorithmus ist ein Verfahren zur Lösung eines gegebenen Problems. Er ist eindeutig, endlich und schrittweise.

2) Die ANSI Norm ist eine Vorschrift, die festlegt, nach welchen Regeln ein C-Programm aufgebaut sein muss.

3) Literale sind Bezeichner mit einem festen Wert wie z.B. 9

4) 2 Bytes kann $2^{16} \approx 64000$ verschiedene Zustände speichern.
Damit können Ganzzahlen von ungefähr -32000 bis ungefähr 32000 abgespeichert werden.
Also kann 29999 in dieser zwei Bytes grossen Integer-Zahl gespeichert werden.

5) Mit fflush(stdin) werden evtl.noch im Tastaturpuffer stehende Zeichen gelöscht.

6)

```
#include "stdafx.h"
#include <stdio.h>

int main(){
	float z1;
	double z2;
	int summe;
	int differenz;
	int produkt;
	int quotient;

	printf("Bitte 1. Zahl eingeben\n");
	scanf("%f",&z1);
	fflush(stdin);

	printf("Bitte 2. Zahl eingeben\n");
	scanf("%lf",&z2);
	fflush(stdin);

	summe = (int)(z1+z2);
	differenz = (int)(z1-z2);
	produkt = (int)(z1*z2);
	quotient = (int)(z1/z2);

	printf("Summe=%d\n", summe);
	printf("Differenz=%d\n", differenz);
	printf("Produkt=%d\n", produkt);
	printf("Quotient=%d\n", quotient);
	return 0;
}
```

7)
```
d = 6/(1/3);
```
`1/3` hat den Wert 0, also wird durch Null dividiert.
Diese Anweisung wird von keinem Mikroprozessor ausgeführt.

8) a)
```
d = (double)2 * 3.14 * r;
```
2 ist ein Integer-Wert. Bei Multiplikation mit einem double-Wert (hier 3.14) wird der Integer-Wert 2 vor der Multiplikation implizit (automatisch, ohne Zutun des Programmierers) in einen double-Wert umgewandelt. Das bedeutet, daß die cast Umwandlung mit (double) nicht gemacht werden muß.

b)
```
i = (int)(4 * 2.718* i)
```
Der Wert von 4 * 2.718* i ergibt einen double-Wert. Wenn man die explizite Umwandlung mit dem cast Operator (int) wegläßt, wird dieser double-Wert in der Integer-Wert Variablen i abgespeichert. Dadurch entsteht ein Datenverlust. Dies meldet der Compiler.

9) Ja.
```
d = 2 * 3.14 * d;
```
Der Wert von 2 * 3.14 * d ergibt einen double-Wert.
Dieser double-Wert wird in der float-Variablen d abgespeichert. Dadurch kann ein Datenverlust entstehen. Dies meldet der Compiler.

KLAUSUR 2 SAE E1FI 12.1.2006 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):
- → Alle Aufgaben müssen bearbeitet werden.
- → Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- → Aufgabenblätter bitte auch abgeben.
- → Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- → Die rote Farbe darf nicht benutzt werden.
- → Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- → Selbsterklärende Variablennamen benutzen.
- → Programme müssen benutzerfreundlich sein.
- → Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- → EVA-Prinzip muss benutzt werden.
- → Einrücken der entsprechenden Programmteile.
- → Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Schreiben Sie ein C-Programm, das den Ersatzwiderstand einer Parallelschaltung von 2 Widerständen berechnet (Datentyp jeweils double).
Wurde ein Widerstandwert kleiner oder gleich Null eingegeben, muß das Programm eine entsprechende Meldung auf den Bildschirm bringen (im Ausgabeteil !).
Dies muß mit dem EVA-Prinzip realisiert werden.

Bemerkung:

$$\frac{1}{R_G} = \frac{1}{R_1} + \frac{1}{R_2}$$

2) Schreiben Sie ein C-Programm, das das Minimum und das Maximum dreier Zahlen (Datentyp jeweils integer) berechnet und auf dem Bildschirm ausgibt.
Es dürfen nur einseitige Verzweigungen benutzt werden!

3) Schreiben Sie ein C-Programm, das die Funktion eines kleinen Taschenrechners besitzt:
Der Anwender muss zuerst zwei Zahlen (Datentyp jeweils integer) eingeben, dann muss er entweder das Zeichen "*" oder irgend ein anderes Zeichen eingeben.
Wenn das Zeichen "*" eingegeben wurde, muss das Produkt berechnet, wenn das Zeichnen "*" nicht eingegeben wurde, muss der Quotient berechnet werden. Das Ergebnis soll als float-Zahl abgespeichert (Nachkommateil nicht abschneiden, Datentyp float) werden.
EVA-Prinzip verwenden !

Lösungen

1)

```
int main(void){
	float r1, r2, r3, rErsatz;
	// Eingabe
	printf("Bitte 1.Widerstandswert eingeben:\n");
	scanf("%f", &r1);
	fflush(stdin);

	printf("Bitte 2.Widerstandswert eingeben:\n");
	scanf("%f", &r2);
	fflush(stdin);

	// Verarbeitung
	if(r1>0 && r2>0){
		rErsatz = 1/r1 + 1/r2;
		rErsatz= 1/rErsatz;
	}

	// Ausgabe
	if(r1>0 && r2>0){
		printf("Ersatzwiderstand = %f\n", rErsatz);
	}
	else{
		printf("Mindestens ein Widerstand hat einen falschen Wert\n");
	}
	return 0;
}
```

2)

```
int main(void){
	int z1, z2, z3, min, max;

	// Eingabe
	printf("Bitte 1. Zahl eingeben:\n");
	scanf("%d", &z1);
	fflush(stdin);

	printf("Bitte 2. Zahl eingeben:\n");
	scanf("%d", &z2);
	fflush(stdin);

	printf("Bitte 3. Zahl eingeben:\n");
	scanf("%d", &z3);
	fflush(stdin);

	// Verarbeitung
	if(z1>z2){
		max=z1;
		min=z2;
	}
	if(!(z1>z2)){
		max=z2;
		min=z1;
	}

	if(max<z3){
		max=z3;
	}

	if(min>z3){
		min=z3;
	}

	// Ausgabe
	printf("Min=%d Max=%d\n",min,max);
	return 0;
}
```

3)

```
int main(){
	int z1;
	int z2;
	char z;
	float erg;

	// Eingabe
	printf("Bitte 1. Zahl eingeben\n");
	scanf("%d",&z1);
	fflush(stdin);

	printf("Bitte 2. Zahl eingeben\n");
	scanf("%d",&z2);
	fflush(stdin);

	printf("Bitte Rechenzeichen * oder / eingeben\n");
	scanf("%c",&z);
	fflush(stdin);

	// Verarbeitung
	if(z=='*'){
		erg=(float)(z1*z2);
	}
	else{
		erg=(float)z1/(float)z2;
	}

	// Ausgabe
	printf("Das Ergebnis ist:= %f\n",erg);

	return 0;
}
```

KLAUSUR 3 SAE E1FI 6.6.2006 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Erstellen Sie ein Struktogramm zu dem Programm, das den Ersatzwiderstand von **maximal** drei parallel geschalteten Widerständen berechnet. Die Widerstände (Einheit: Ohm) müssen über Tastatur eingegeben werden. Wenn für einen Widerstand ein Wert kleiner oder gleich 0 eingegeben wird, wird der Ersatzwiderstand der bisher eingegebenen Widerstände berechnet und ausgegeben (bzw. eine entsprechende Meldung, wenn kein Ersatzwiderstand existiert). Danach wird das Programm **beendet** (ohne auf eine weitere Eingabe zu warten).
D.h. es kann kein Ersatzwiderstand bzw. der Ersatzwiderstand von 1, 2 oder 3 parallelen Widerständen berechnet werden.

2) Erstellen Sie ein Struktogramm zu dem folgenden Programm:
Es soll ein Taschenrechner (er soll die 4 Grundrechenarten beherrschen) programmiert werden.

a/A: Berechnet die Summe
b/B: Berechnet die Differenz
c/C: Berechnet das Produkt
d/D: Berechnet den Quotient
sonst: Programmende

Bitte EVA-Prinzip beachten!
Bei Division durch Null entsprechende Meldung bringen und dann das Programm sofort (ohne Rechnung) beenden.

KLAUSUR 4 SAE E1FI 29.6.2006 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):
- → Alle Aufgaben müssen bearbeitet werden.
- → Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- → Aufgabenblätter bitte auch abgeben.
- → Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- → Die rote Farbe darf nicht benutzt werden.
- → Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- → Selbsterklärende Variablennamen benutzen.
- → Programme müssen benutzerfreundlich sein.
- → Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- → EVA-Prinzip muss benutzt werden.
- → Einrücken der entsprechenden Programmteile.
- → Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Was ist allgemein der Hauptunterschied zwischen einer While-Anweisung und einer do-while-Anweisung ?
Geben Sie ein **konkretes** Beispiel (Programmteil) an, bei dem bei gleicher Schleifenbedingung und gleichem Schleifenrumpf eine do-while-Anweisung etwas anderes macht, als eine while-Anweisung.

2) Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
i=10;
while(i<20)
  printf("Hallo Welt\n");
```

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?
Begründen Sie !

3) Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
i=20;
while(i<20)
  printf("Hallo Welt\n");
```

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?
Begründen Sie !

4) `// Programmausschnitt P1`

```
        sum = 0;
        i = 1;

    --> while ... {
          sum = sum +i;
          i = i+1;
          }
```

a) Tragen Sie insgesamt vier Mal die Werte der Variablen i und sum beim Erreichen des Programms an der mit --> gekennzeichneten Stelle in die Tabelle ein (unter der Annahme, dass die mit ... bezeichnete Schleifenbedingung wahr ist).

i					
sum					

b) Ergänzen Sie den Programmausschnitt P1 – durch die Angabe einer konkreten Schleifenbedingung – so, dass der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe 1 + 2 + 3 + 4 + 5 ist.

c) Verändern Sie den Programmausschnitt P1 – nur durch die Angabe einer anderen Initialisierung und einer konkreten Schleifenbedingung – so, daß die Summe
5 + 6 + 7 + 8
berechnet wird. Tragen Sie die Werte für sum und i wieder in eine entsprechende Tabelle ein

5)
```
        // Programmausschnitt P2
        sum = 0;
        i = 0;

    --> while ... {
          i = i+1;
          sum = sum +i;
          }
```

a) Tragen Sie insgesamt vier Mal die Werte der Variablen i und sum beim Erreichen des Programms an der mit --> gekennzeichneten Stelle in die Tabelle ein (unter der Annahme, dass die mit ... bezeichnete Schleifenbedingung wahr ist).

i					
sum					

b) Ergänzen Sie den Programmausschnitt P2 – durch die Angabe einer konkreten Schleifenbedingung – so, dass der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe 1 + 2 + 3 + 4 + 5 ist.

c) Verändern Sie den Programmausschnitt P2 – nur durch die Angabe einer anderen Initialisierung und einer konkreten Schleifenbedingung – so, daß die Summe
5 + 6 + 7 + 8
berechnet wird. Tragen Sie die Werte für sum und i wieder in eine entsprechende Tabelle ein

Lösungen:

1) Anzahl Durchgänge der do-while-Anweisung: >= 1
Anzahl Durchgänge der while-Anweisung: >= 0
while-Anweisung: vor dem Schleifenrumpf wird Bedingung überprüft
do-while-Anweisung: nach dem Schleifenrumpf wird Bedingung überprüft

```
i=10;
do{
  printf("%d ",i);
}
while(i<10);
```

```
i=10;
while(i<10){
  printf("%d ",i);
}
while(i<10);
```

6P

2)
Unendlich oft, weil die Bedingung immer wahr ist.

3P

3)
Null Mal, weil die Bedingung falsch ist.

3P

4)
a)

i	1	2	3	4	5
sum	0	1	1+2	1+2+3	1+2+3+4

5P

b)
while (i<=5)

5P

c)

```
            sum = 0;
            i = 5;

        --> while{ (i<=8)
              sum = sum +i;
              i = i+1;
              }
```

Tabelle:

i	5	6	7	8	9
sum	0	5	5+6	5+6+7	5+6+7+8

9P

5)

a)

i	0	1	2	3	4
sum	0	1	1+2	1+2+3	1+2+3+4

5P

b)

while (i<5)

5P

c)

```
            sum = 0;
            i = 4;

        --> while{ (i<8)
              i = i+1;
              sum = sum +i;
              }
```

Tabelle:

i	4	5	6	7	8
sum	0	5	5+6	5+6+7	5+6+7+8

9P

KLAUSUR 2 SAE E1FI Nachtermin1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

Bemerkungen:
Bitte EVA-Prinzip so weit - wie möglich - einhalten !!

1) Schreiben Sie ein C-Programm, in dem 10 positive Zahlen über Tastatur eingegeben werden.
Das Programm soll die kleinste Zahl ermitteln und auf dem Bildschirm ausgeben.
Benutzen Sie dazu eine Schleife.

2) Schreiben Sie ein C-Programm, das den Mittelwert von Zahlen berechnet.
Der Anwender muß zuerst die Anzahl der Zahlen und dann die einzelnen Zahlen über Tastatur eingeben.
Der Mittelwert soll dann über Tastatur ausgegeben werden.

KLAUSUR 1 SAE E2FI 20.11.2006 Zeit: 60 Minuten

Name, Vorname:

Hilfsmitttel:
Bücher und Skripte eigener Wahl.

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Die Lösungen müssen als ablauffähige C-Programme dargestellt werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mitaufgenoommen werden.
→ Die rote Farbe darf nicht benutzt werden.
→ Aufgabenblätter bitte auch abgeben.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Die Programme müssen ausgedruckt werden.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

Bemerkung:

1) Es dürfen keine nicht reservierten Zellen eines Feldes überschrieben werden.
2) Die Feldlänge soll vom Programmierer ohne aufwändige Bemühungen an genau einer Stelle

leicht zu verändern sein.(und muß dann neu compiliert werden) und soll zuerst einmal den Wert 5 haben.

1) Geben Sie die syntaktischen Fehler und die Fehler, die zur Laufzeit entstehen können, an. (6 P)

```
#include "stdafx.h"
#include <stdio.h>

int main(){
     int i = 3;
     double w[i];
     int v[3]={10,9,15,8};

     v[2] = 10;
     v[i+1]=v[2];
     printf("r=%d\n",v[i-3]);
     return 0;
}
```

2)

a) Schreiben Sie ein C-Programm, das die Umsätze (grösser Null) einer Firma in ein Feld einliest. Die Umsätze müssen über Tastatur eingegeben werden. Wenn für einen Umsatz ein Wert kleiner als Null eingegeben wird, wird die Eingabe beendet. Das Ende der Umsätze im Feld muss durch einen negativen Wert dargestellt sein. (7P)

b) Geben Sie diese Umsätze auf dem Bildschirm (beginnend mit der nullten Zelle des Feldes) aus. (4P)

c) Kopieren Sie die Umsatzzahlen des Feldes in ein anderes Feld. (4P)

Lösungen:
1)
a) double w[i];
Feldlänge muss eine Konstante sein.
b) int v[3]={10,9,15,8};
Zu viele Initialisierungen (Feldlänge ist nur 3)
c) v[i+1]=v[2];
Da i den Wert 4 hat, greift das Programm auf nichtreservierten Speicher zu.

2)

```
#include <stdafx.h>
#include <stdio.h>

const int len = 3;

int main(){
     double umsatzArray[len];
     double kopieUmsatzArray[len];
     int i=0;
     int beenden=0; //nicht Schleife beenden

     // 2a)
     do{
          printf("Bitte Umsatz eingeben: ");
          scanf("%lf", &umsatzArray[i]);
          if(umsatzArray[i]<0){
               beenden = 1;
          }
          else if (i==len-1){
               umsatzArray[len-1]=-1;
               beenden = 1;
          }
          i=i+1;
     } while(beenden==0);

     // 2b)
     i=0;
     do{
          printf("%f ", umsatzArray[i]);
          i=i+1;
     } while(umsatzArray[i]>=0);

     // 2c)
     i=0;
     do{
          kopieUmsatzArray[i]=umsatzArray[i];
          i=i+1;
     } while(umsatzArray[i]>=0);
     kopieUmsatzArray[i]=-1;

     return 0;
}
```

KLAUSUR 1 SAE E1FI 7.12.2006 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Welche Eigenschaften hat ein Algorithmus ? (1 P)

2) Was ist die ANSI Norm ? (1 P)

3) Kann die Ganzzahl 29999 in einer zwei Bytes grossen Integer-Zahl abgespeichert werden ? Begründen Sie! (1 P)

4) Warum ist folgende Anweisung nach Verwendung der scanf-Funktion sinnvoll ?
`fflush(stdin);` (1 P)

5) a) Wie viele Zustände genau kann man mit 4 Byte darstellen ? (1 P)
Geben Sie nur die Formel an.
b) Wie kann man dies näherungsweise ausrechnen (wie im Unterricht) ? (1 P)

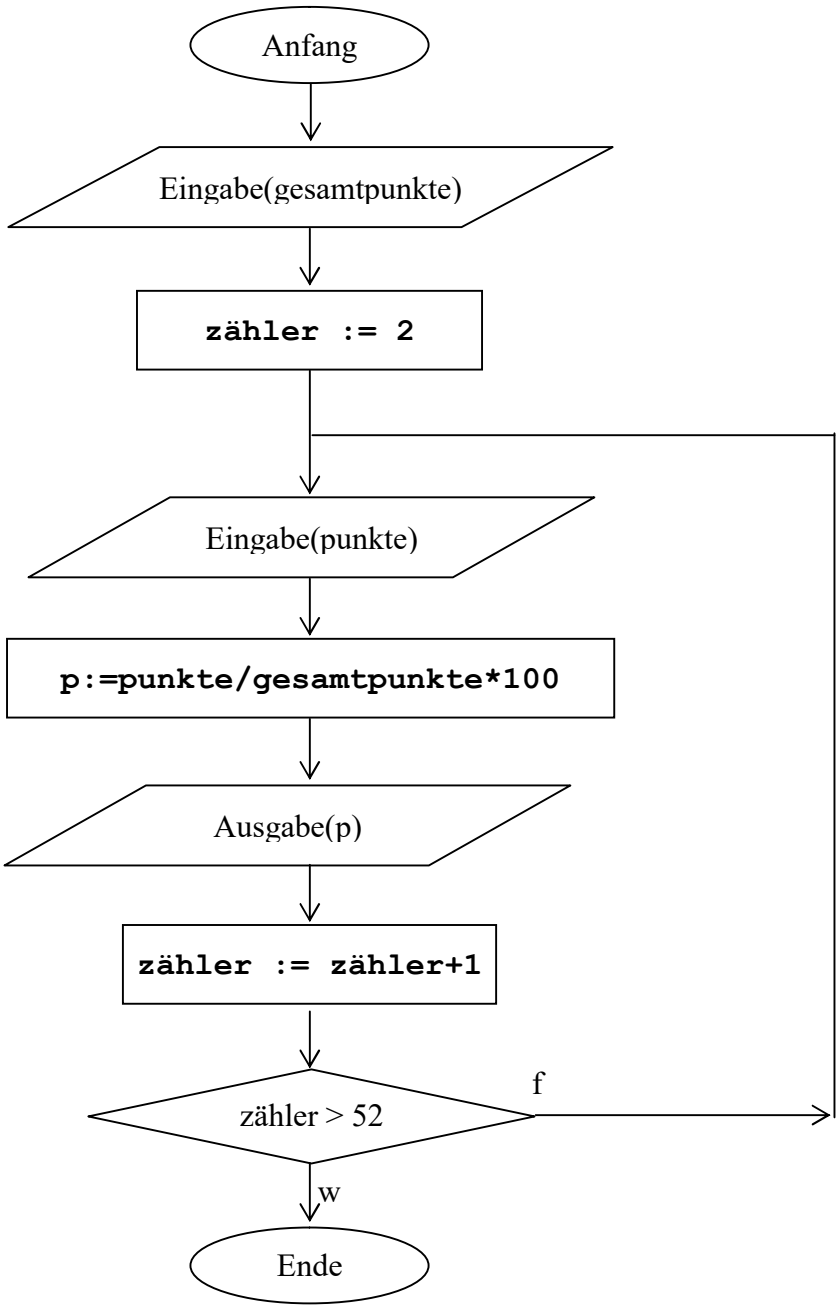
6) Schreiben Sie ein syntaktisch korrektes C-Programm, das den Ersatzwiderstand von drei parallel geschalteten Widerständen berechnet. (8 P)

$$\frac{1}{R_{ers}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$$

7)
Es soll von 50 Schülern jeweils der Prozentsatz der erreichten Punkte bzgl. der Gesamtpunktzahl in einer Klassenarbeit errechnet und ausgegeben werden.

a) Ist die folgende Lösung korrekt ? Begründen Sie mit der Tabelle unten! (3 P)
b) Falls sie nicht korrekt ist, verbessern sie die Lösung. (3 P)
Suchen Sie dazu eine geeignete Stelle im Programm aus (**markieren** diese mit einem Pfeil), an der Sie das Programm gedanklich bei jedem Schleifendurchgang anhalten und dort den Wert der Variablen zähler und die Anzahl der ausgegebenen Schüler protokollieren:

Anzahl der ausgegebenen Schüler	Wert der Variablen zähler



Lösungen

1) Ein Algorithmus ist ein Verfahren zur Lösung eines gegebenen Problems. Er ist eindeutig, endlich und schrittweise.

2) Die ANSI Norm ist eine Vorschrift, die festlegt, nach welchen Regeln ein C-Programm aufgebaut sein muss.

3) 2 Bytes kann $2^{16} \approx 64000$ verschiedene Zustände speichern.
Damit können Ganzzahlen von ungefähr -32000 bis ungefähr 32000 abgespeichert werden.
Also kann 29999 in dieser zwei Bytes grossen Integer-Zahl gespeichert werden.

4) Mit fflush(stdin) werden evtl.noch im Tastaturpuffer stehende Zeichen gelöscht.

5) a) 2^{32}

b) $2^{32} = 2^{2+30} = 2^2 \cdot 2^{30} = 4 \cdot 2^{30} = 4 \cdot 2^{10 \cdot 3} = 4 \cdot (2^{10})^3 \approx 4 \cdot 1000^3 \approx 4 \cdot 1000000000$

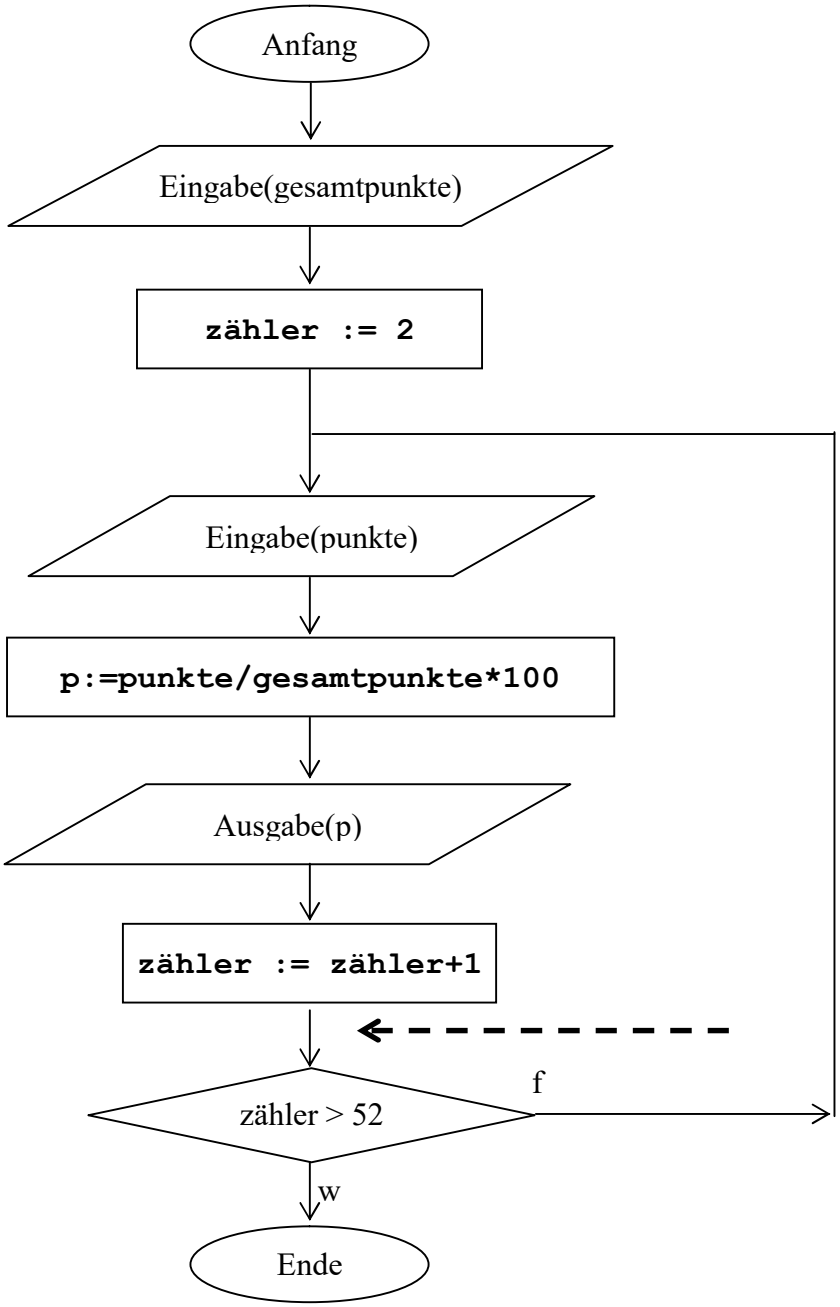
6) Siehe Unterricht

7a)

Anzahl der ausgegebenen Schüler	Wert der Variablen zähler
1	3
2	4
...	...
52 – 2 = 50	52

Zusammenhang:
anzahl = zähler – 2
also: Lösung korrekt

b) Keine Korrektur nötig



KLAUSUR 2 SAE E1FI 11.1.2007 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):
→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) Was bedeutet Priorität in der Programmiersprache C ? (2P)
Geben Sie ein Beispiel.

2) Was bedeutet Assoziativität in der Programmiersprache C ? (2P)
Geben Sie ein Beispiel.

3) Welche Werte haben die folgenden Ausdrücke? (14P)
Bemerkung:
- Die Variable x ist mit 10 und die Variablen a, b, c sind jeweils mit 0 initialisiert (vorbelegt).
- Der Operator == ist rechtsassoziativ

Sind die folgenden Ausdrücke syntaktisch korrekt?
Wenn ja, geben Sie die Werte der Ausdrücke an.

a) `(x=9)-1`
b) `a==b==c`
c) `7-2*2==(1&&2)`
d) `x=(2&&8)`
e) `(a||b)+12`
f) `(x=2)&&8`
g) `a&&b||c`

4) Welche der folgenden Zeichenreihen sind **syntaktisch korrekte** Ausdrücke (in der Programmiersprache C) und welche **Werte** haben diese Ausdrücke ? (33P)
Bem:
Die Variablen sollen alle den Wert 123 haben.

Ausdruck	Wert
`3 = 2+7`	
`3 * 3 + 4 * 4 == 5 * 5`	
`x = 7 /*8+34`	
`2x = x + x`	
`y == y+1`	
`7 && 8`	
`7 || 8`	
`!23`	
`!7 || !8`	
`x!=5`	
`x==!5`	
`i=(3==5)`	
`j!=(4==4)`	

5) Formulieren Sie die folgenden mathematischen Ausdrücke als syntaktisch korrekte Ausdrücke in der Programmiersprache C.

`z = 2+6/3`	
$4 \leq x$	3 Möglichkeiten
$2 < x < 5$	
$a \leq x < 8$	
$1 < x \leq 7$	
$b \leq x \leq 9$	
$6 \geq x$	
$d > x > r$	
$a \geq x \geq 83$	
$1 > x \geq 17$	
$b \not\geq x \geq 9.4$	
$y \not> x$	
$b \not\geq x$	
$u \not< 3.14$	
$a = b = c$	
$x \neq 5$	3 Möglichkeiten

Lösung:

1) Priorität gibt den Vorrang eines Operators in einem Ausdruck an.
Beispiel: Punkt vor Strich.

2) Assoziativität gibt an, in welcher Reihenfolge (von links nach rechts oder von rechts nach links) Operatoren mit der gleichen Priorität abgearbeitet werden.
Beispiel: * wird von links nach rechts abgearbeitet.

3)

a)	`(x=9)-1`	Wert 8
b)	`a==b==c`	Wert 0
c)	`7-2*2==(1&&2)`	Wert 0
d)	`x=(2&&8)`	Wert 1
e)	`(a||b)+12`	Wert 12
f)	`(x=2)&&8`	Wert 1
g)	`a&&b||c`	Wert 0

4)

Ausdruck	Wert
`3 = 2+7`	syntaktisch inkorrekt
`3 * 3 + 4 * 4 == 5 * 5`	1
`x = 7 /*8+34`	syntaktisch inkorrekt
`2x = x + x`	syntaktisch inkorrekt
`y == y+1`	0
`7 && 8`	1
`7 || 8`	1
`!23`	0
`!7 || !8`	0
`x!=5`	1
`x==!5`	0
`i=(3==5)`	0
`j!=(4==4)`	1

5)

`z = 2+6/3`	`z==2+6/3`
`4 ≤ x`	`4<=x  4<x||4==x  !(x<3)`
`2 < x < 5`	`2<x && x<5`
`a ≤ x < 8`	`a<=x && x<8`
`1 < x ≤ 7`	`1<x && x<=7`
`b ≤ x ≤ 9`	`b<=x && x<=9`
`6 ≥ x`	`6>=x`
`d > x > r`	`d>x && x>r`
`a ≥ x ≥ 83`	`a>=x && x>=83`
`1 > x ≥ 17`	`1>x && x>=17`
`b ≱ x ≥ 9,4`	`!(b>=x) && x>=9.4`
`y ≯ x`	`!(y>x)`
`b ≱ x`	`!(b>=x)`
`u ≮ 3,14`	`!(u<3.14)`
`a = b = c`	`a==b && b==c`
`x ≠ 5`	`!(x==5)  x!=5  x<5||x>5`

KLAUSUR 3 SAE E1FI 16.4.2007 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

Bemerkung:
Alle für eine Funktion "wichtigen" Informationen müssen als Parameter realsiert werden, nicht als lokale Variablen.

1) Schreiben Sie eine Funktion, die das Volumen eines Quaders berechnet.
Rufen Sie die Funktion im Hautpprogramm auf.

2) Schreiben Sie ein Funktion, die den Benzinverbrauch pro 100 km berechnet.
Erzeugen Sie dazu auch einen Aufruf im Hauptprogramm.

3) Schreiben Sie ein Funktion, die (in Abhängikeit eines weiteren Parameters) aus einem Bruttopreis und der Mehrwertsteuer den Nettopreis, bzw. aus das einem Nettopreis (und der Mehrwertsteuer) den Bruttopreis berechnet.

4) Schreiben Sie ein C-Programm, das die Funktion eines kleinen Taschenrechners besitzt:
Der Anwender muss zuerst zwei Zahlen (Datentyp jeweils integer) eingeben, dann muss er entweder das Zeichen "*" oder irgend ein anderes Zeichen eingeben.
Wenn das Zeichen "*" eingegeben wurde, muss das Produkt berechnet, wenn das Zeichnen "*" nicht eingegeben wurde, muss der Quotient berechnet werden. Das Ergebnis soll als float-Zahl abgespeichert (Nachkommateil nicht abschneiden, Datentyp float) werden.
EVA-Prinzip verwenden !

KLAUSUR 3 SAE E1FI 26.4.2007 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):
- → Alle Aufgaben müssen bearbeitet werden.
- → Der Name und Vorname muß auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
- → Aufgabenblätter bitte auch abgeben.
- → Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
- → Die rote Farbe darf nicht benutzt werden.
- → Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
- → Selbsterklärende Variablennamen benutzen.
- → Programme müssen benutzerfreundlich sein.
- → Programme dürfen beim Kompilieren keine Warnungen und keine Fehlermeldungen bringen.
- → EVA-Prinzip muss benutzt werden.
- → Einrücken der entsprechenden Programmteile.
- → Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) a) Es soll über Tastatur eine **ganze** Zahl eingegeben werden. (20P)
Ist diese Zahl eine Schulnote zwischen 1 und 4 (je einschließlich) wird die Meldung "Prüfung bestanden" ausgegeben.
Ist diese Zahl die Schulnote 5 oder 6 wird die Meldung "Prüfung nicht bestanden" ausgegeben.
Sonst wird die Meldung "Diese Zahl ist keine Note" ausgegeben.
Realisieren Sie dies durch ein C-Programm (Programmausschnitt) nur mit switch-Anweisung, die aus möglichst wenig Anweisungen besteht
(keine if-else-Anweisung, keine if-Anweisung)
b) Erklären Sie die if-else-if-Ketten an einem Beispiel (Programmauschnitt).
c) Gibt es ein Problem, das man mit einer if-else-Anweisung, aber nicht mit einer switch-Anweisung lösen kann? (gibt es so einen Fall?)
d) Gibt es ein Problem, das man mit einer switch-Anweisung, aber nicht mit einer if-else-Anweisung lösen kann? (gibt es so einen Fall?)

2) Es soll der Ersatzwiderstand einer Parallelschaltung von 3 Widerständen berechnet werden.
Wurde ein Widerstandwert kleiner oder gleich Null eingegeben, muß das Programm sofort beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden.
(insbesondere darf dann nicht mehr ein weiterer Widerstand eingegeben und der Ersatzwiderstand berechnet werden).
Dies muß mit dem EVA-Prinzip realisiert werden.
Außerdem muß in der Ausgabe noch angegeben werden, welcher Widerstandswert (z.B. 1. Widerstandwert) falsch eingegeben wurde.
Erstellen Sie das dazugehörige Struktogramm. (30P)

Bemerkungen:
a) $\frac{1}{R_G} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}$

b) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob der 3. Widerstandswert kleiner oder gleich 0 ist) und diese dann im Ausgabeteil abgeprüft werden.

Lösungen
1)
a)
```
int main(){
  int note;
  printf("Bitte eine ganze Note eingeben\n");
  scanf("%c", &note);

  switch(note){   // int oder char
    case 1:
    case 2:
    case 3:
    case 4:
      printf("Prüfung bestanden \n");
      break;      // nicht abweisend

    case 5:
    case 6:
      printf("Prüfung nicht bestanden \n");
      break;

    default:   // falls kein case-Fall zutrifft
      printf("Dies ist keine Note \n");
      break;
  }
  return 0;
}
```

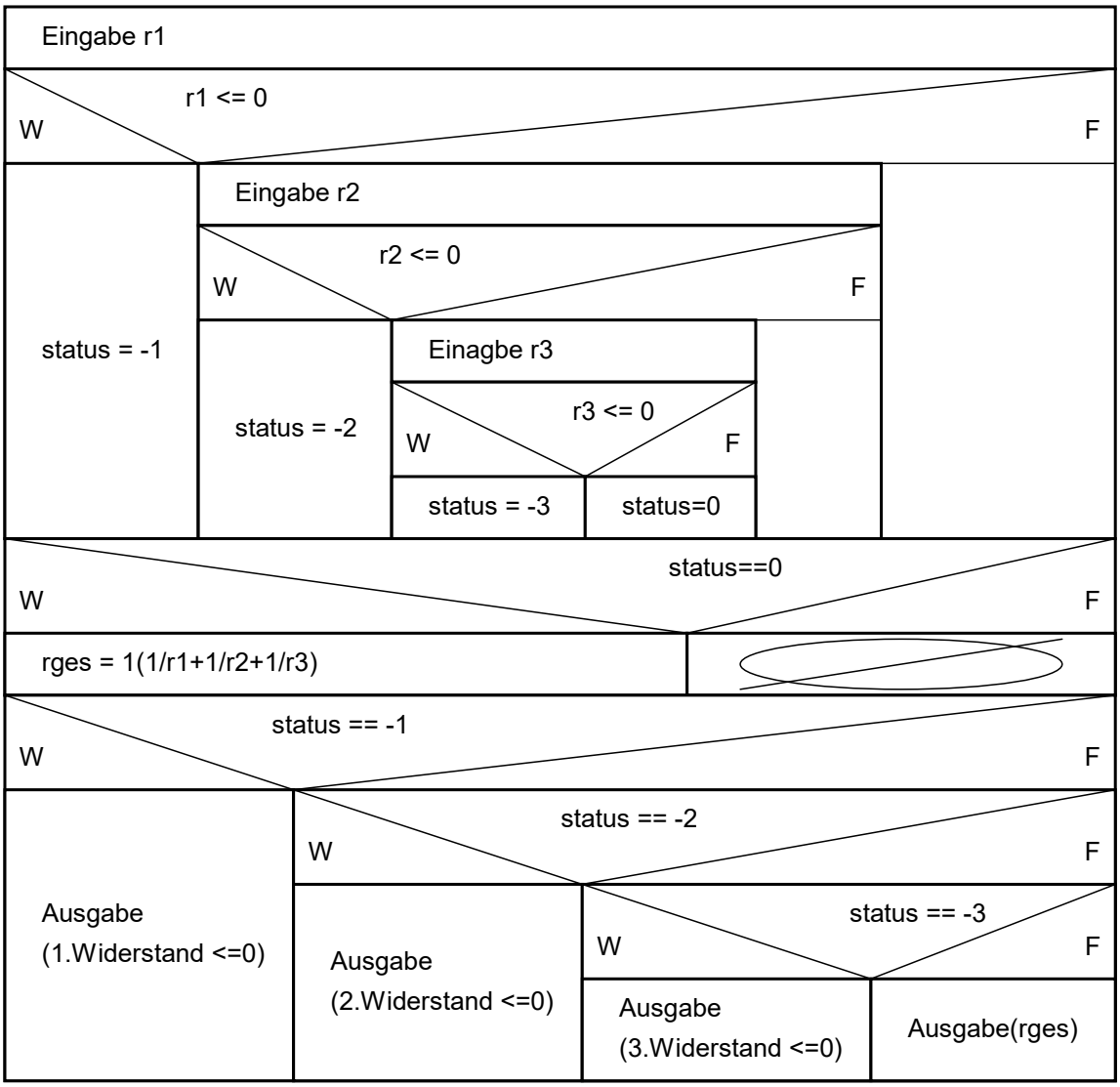
b)
```
int main(){
  int note;
  printf("Bitte eine ganze Note eingeben\n");
  scanf("%c", &note);

  if(note==1){
    printf("Note sehr gut\n");
  else if(note==2){
    printf("Note gut\n");
  else if(note==3){
    printf("Note befriedigend\n");
  else if(note==4){
    printf("Note ausreichend\n");
  else if(note==5){
    printf("Note mangelhaft\n");
  else if(note==6){
    printf("Note ungenügend\n");
  else{
    printf("Dies ist keine Note\n");
}
```

c)
switch: nur für ganzzahlige oder char Datentypen verwendbar

d) nichts

2)

Widerstandsberechnung



Ersatzaufgabe

1) Der Anwender muss in einem Programm eine ganze Zahl eingeben. Wenn die Zahl zwischen 1 und 12 ist, soll sie den Monat in einem Jahr repräsentieren (z.B. 2 den Februar). Schreiben Sie ein Programm, das bei Eingabe einer Zahl zwischen 1 und 12 diese als Monat auffast und ausgibt, zu welcher Jahreszeit dieser Monat gehört (Frühling, Sommer, Herbst, Winter). Es soll wie immer angenommen werden, dass der Anwender Eingabefehler machen kann.

KLAUSUR 4 SAE E1FI 14.6.2007 Zeit: 45 Minuten

Name, Vorname:

Hilfsmitttel:
keine

Hinweise (unbedingt beachten):

→ Alle Aufgaben müssen bearbeitet werden.
→ Der Name und Vorname muß in DRUCKSCHRIFT auf jedes Aufgabenblatt und auf jedes Lösungsblatt geschrieben werden.
→ Aufgabenblätter bitte auch abgeben.
→ Die Lösungsblätter müssen in folgender Form durchnummeriert werden. Beispiel: 1/4 2/4 3/4 4/4
→ Die rote Farbe darf nicht benutzt werden.
→ Lassen Sie bitte auf der linken Seite einen mindestens 3cm breiten Rand.
→ Selbsterklärende Variablennamen benutzen.
→ Programme müssen benutzerfreundlich sein.
→ EVA-Prinzip muss benutzt werden.
→ Einrücken der entsprechenden Programmteile.
→ Sichtbarkeits- und Zugriffsmodifizierer wie public, usw. dürfen nicht weggelassen werden.
→ In allen Klassen dürfen in keiner Methode Ein- oder Ausgaben (auf dem Bildschirm) vorkommen, außer es handelt sich um reine Ausgabe- oder Eingabe Methoden.
→ Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!

AUFGABEN

1) 6P

Was ist allgemein der Hauptunterschied zwischen einer While-Anweisung und einer do-while-Anweisung ?
Geben Sie ein **konkretes** Beispiel (Programmteil) an, bei dem bei gleicher Schleifenbedingung und gleichem Schleifenrumpf eine do-while-Anweisung etwas anderes macht, als eine while-Anweisung.

2) 3P

Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
i=10;
while(i<20)
  System.out.println("Hallo Welt\n");
```

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?
Begründen Sie !

3) 3P

Gegeben ist der folgende syntaktisch korrekte Programmteil:

```
i=20;
while(i<20)
  System.out.println("Hallo Welt\n");
```

Wie oft gibt dieser Programmteil die Meldung "Hallo Welt" auf dem Bildschirm aus ?
Begründen Sie !

4) 19P

```
        // Programmausschnitt P1
         sum = 0;
         i = 1;

--> while ... {
       sum = sum +i;
       i = i+1;
       }
```

a) Tragen Sie insgesamt vier Mal die Werte der Variablen i und sum beim Erreichen des Programms an der mit --> gekennzeichneten Stelle in die Tabelle ein (unter der Annahme, dass die mit ... bezeichnete Schleifenbedingung wahr ist).

i					
sum					

b) Ergänzen Sie den Programmausschnitt P1 – durch die Angabe einer konkreten Schleifenbedingung – so, dass der Wert der Variablen sum nach Verlassen der Schleife gleich der Summe 1 + 2 + 3 + 4 + 5 ist.

c) Verändern Sie den Programmausschnitt P1 – nur durch die Angabe einer anderen Initialisierung und einer konkreten Schleifenbedingung – so, daß die Summe
5 + 6 + 7 + 8
berechnet wird. Tragen Sie die Werte für sum und i wieder in eine entsprechende Tabelle ein

5) 19P

Schreiben Sie ein Programm, das die Zahl n^n einer über Tastatur eingegebenen ganzen Zahl n ≥ 1 bestimmt.

Beispiel:
n = 3
$n^n = 3^3 = 3 \cdot 3 \cdot 3 = 27$

Lösungen:

1) Anzahl Durchgänge der do-while-Anweisung: >= 1
Anzahl Durchgänge der while-Anweisung: >= 0
while-Anweisung: vor dem Schleifenrumpf wird Bedingung überprüft
do-while-Anweisung: nach dem Schleifenrumpf wird Bedingung überprüft

```
i=10;
do{
  printf("%d ",i);
}
while(i<10);
```

```
i=10;
while(i<10){
  printf("%d ",i);
}
while(i<10);
```

6P

2)
Unendlich oft, weil die Bedingung immer wahr ist.

3P

3)
Null Mal, weil die Bedingung falsch ist.

3P

4)
a)

i	1	2	3	4	5
sum	0	1	1+2	1+2+3	1+2+3+4

5P

b)
while (i<=5)

5P

c)

```
            sum = 0;
            i = 5;

        --> while{ (i<=8)
              sum = sum +i;
              i = i+1;
              }
```

Tabelle:

i	5	6	7	8	9
sum	0	5	5+6	5+6+7	5+6+7+8

9P

5) 19P