

*Name, Vorname:*

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1) 3+4+4 P

- a) Welche 3 Möglichkeiten (3 Worte nennen) gibt es, einen Algorithmus darzustellen?
- b) Erklären Sie die Begriffe Syntax und Semantik?
- c) Was ist ein Compiler ?

2) 5 P

- a) Wie viele Zustände genau kann man mit 8 Byte angeben? (als Hochzahl schreiben)
- b) Näherungsweise Berechnung!

3) 14P

Schreiben Sie ein C- Programm, das die größte von 3 über Tastatur eingegebenen, ganzen Zahlen z1, z2, z3 bestimmt und in der Variablen max speichert. Ein- und Ausgabebeteil wird nicht verlangt!

4) 10P

Das folgende syntaktisch korrekte C-Programm soll die kleinste Zahl (dreier Zahlen) berechnen und auf dem Bildschirm ausgeben.

Wenn das Programm korrekt ist, schreiben Sie "Der Algorithmus ist korrekt" und geben 5 Beispiele (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von min) an, wo er korrekt wird.

Wenn das Programm nicht korrekt ist, schreiben Sie " Der Algorithmus ist nicht korrekt" und geben ein **konkretes** Beispiel (mit konkreten Werten für z1, z2, z3 und dem berechneten Wert von min) an, bei dem der Algorithmus falsch wird.

Geben Sie dazu genau an, wie der Fluß (Wert der Variablen in das Programm eintragen) durch das Programm verläuft (einzeichnen).

```

int main(){
    int z1,z2,z3, min;
    printf("1.Zahl eingeben:\n");
    scanf("%d",&z1);
    printf("2.Zahl eingeben:\n");
    scanf("%d",&z2);
    printf("3.Zahl eingeben:\n");
    scanf("%d",&z3);
    min = 0;
    if(z1<z2){
        min=z1;
    }
    if(z3<min){
        min=z3;
    }
    printf("Minimum = %d\n",min);
    return 0;
}

```

5)

12P

Durch den folgenden Algorithmus (Flußdiagramm) soll die Summe

$3 + 4 + 5 + 6 + \dots + 100$  berechnet werden.

Dazu wird das Programm immer wieder an der mit <--- bezeichneten Stelle vor der Verzweigung (bei jedem Schleifendurchgang) gedanklich angehalten (Protokoll) und die Werte der Variablen i und sum protokolliert.

a) Tragen Sie dazu in der Tabelle unten die Werte von i und sum bei den ersten 5 Durchgängen ein. Bitte sum **nicht** berechnen, sondern die Summe jeweils darstellen, wie z.B.  $7 + 9 + 11$ .

b) Welche Werte haben i und sum, wenn das Programm das **letzte** Mal an die mit <--- bezeichnete Stelle kommt?

c) Ist das Programm semantisch korrekt (d.h. wird also durch das Programm die Summe  $3 + 4 + 5 + 6 + \dots + 100$  berechnet). Bitte Begründung angeben!

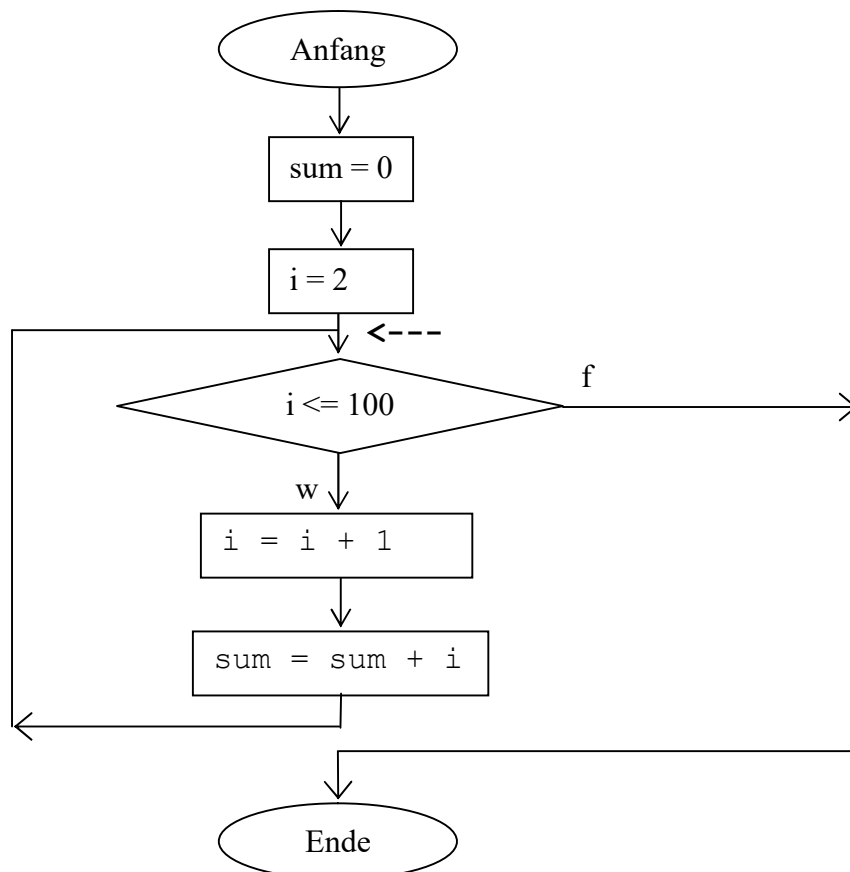


Tabelle (Protokoll):

i							
sum							

Lösung:

1)

a)

3P

Flussdiagramm, Struktogramm, Programm

b)

4P

Die Syntax definiert die äußeren Formgesetze dieser Programmiersprache (ähnlich den grammatikalischen Regeln einer natürlichen - wie z.B. der englischen- Sprache).

Die Semantik ist der Bedeutungsinhalt (ähnlich der Bedeutung der einzelnen Worte einer natürlichen - wie z.B. der italienischen - Sprache) der einzelnen Objekte einer Programmiersprache.

c)

4P

Ein Compiler ist ein Übersetzer, der einen in einer höheren Programmiersprache formulierten Text (ein sogenanntes Programm) in einen aus Maschinenbefehlen bestehenden Text (einem sogenannten Maschinenprogramm) verwandelt. Dieses kann dann vom Mikroprozessor abgearbeitet (ausgeführt) werden.

2)

5 P

$$2^{64} = 2^{60+4} = 2^{60} \cdot 2^4 = 16 \cdot 2^{60} = 16 \cdot 2^{10 \cdot 6} = 16 \cdot (2^{10})^6 \approx 16 \cdot (10^3)^6 = 16 \cdot 10^{18}$$

3)

14P

```
...
if (z1 <= z2) {
    max = z2;
}
else {
    max = z1;
}
if (max < z3) {
    max = z3;
}
...
```

4)

10P

Der Algorithmus ist nicht korrekt:

$$z1 = 30 \quad z2 = 20 \quad z3 = 10 \implies \min = 0$$

5)

12P

a)

5P

i	2	3	4	5	6	...	101
sum	0	0+3=3	3+4	3+4+5	3+4+5+6	...	3+...+101

b)

4P

$$i = 101 \text{ und } \text{sum} = 3 + \dots + 101$$

c)

3P

Da nach das Programm an dieser Stelle beendet wird, haben i und sum die bei b) protokollierten Werte. Das Programm ist also nicht korrekt.

# KLAUSUR 1 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1) 10 P

Simulieren Sie die folgende if-else Verzweigung durch eine oder mehrere einseitige Verzweigungen, wobei außer dem nicht Operator ! keine weiteren logischen Operatoren verwendet werden dürfen.

```
if (B1 && B2) {  
    A1  
}  
else {  
    A2  
}
```

2) 20 P

a) Schreiben Sie ein C-Programm (nur den Verarbeitungsteil), das von 2 Mengen A und B (die jeweils aus ganzen Zahlen bestehen) mit

$A = \{a_1, a_2\}$  mit  $a_1 \neq a_2$

$B = \{b_1, b_2\}$  mit  $b_1 \neq b_2$

den Durchschnitt bestimmt.

Der Durchschnitt sind genau die Zahlen, die sowohl in A als auch in B vorkommen.

b) Erstellen Sie dazu ein Testprotokoll mit 5 verschiedenen Tests.

3) 20 P

Eine natürliche Zahl  $n$  ( $n \geq 2$ ) heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist.

Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13, ....

a) Erstellen Sie ein Flussdiagramm, das folgendes macht:

Es muß von einer eingegebenen ganzen Zahl  $z \geq 2$  festgestellt werden, ob diese eine Primzahl ist. Das Ergebnis muß dann auf dem Bildschirm ausgegeben werden.

b) Erstellen Sie dazu ein Testprotokoll mit 5 verschiedenen Tests.

# KLAUSUR 1 Programmierpraktikum 2BK11 9.12.2014 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Fehler erzeugen !!!

## AUFGABEN

1) 50P

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 C-Programm

Bemerkungen:

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

2) Dies muß mit dem **EVA-Prinzip** realisiert werden.

3) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob das Programm beendet werden soll) und diese dann im Ausgabeteil abgeprüft werden.

4) return darf nur einmal (am Ende des Programms) benutzt werden.

Programm darf nicht durch exit(...) oder sonstige Befehle verlassen bzw. beendet werden.

Lösung:

```
#include "stdafx.h"
int main()
{
    double r1, r2, r3, ersatz;
    int zustand=0;
    // Eingabe
    printf("Eingabe 1. Widerstand\n");
    scanf("%lf",&r1);
    if(r1<=0)
        zustand=-1;
    if(zustand==0){
        printf("Eingabe 2. Widerstand\n");
        scanf("%lf",&r2);
        if(r2<=0)
            zustand=-2;
    }
    if(zustand==0){
        printf("Eingabe 3. Widerstand\n");
        scanf("%lf",&r3);
        if(r3<=0){
            zustand=-3;
        }
    }

    // Verarbeitung
    if(zustand==0){
        ersatz=1/(1/r1+1/r2+1/r3);
    }

    // Ausgabe
    if(zustand==0){
        printf("ersatz=%f\n",ersatz);
    }
    else{
        if(zustand==-1){
            printf("1. Widerstandwert falsch");
        }
        else{
            if(zustand==-2){
                printf("2. Widerstandwert falsch");
            }
            else{
                printf("3. Widerstandwert falsch");
            }
        }
    }

    return 0;
}
```

# KLAUSUR 1 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1) Schreiben Sie ein C-Programm, das die Vereinigungsmenge zweier Zahlenmengen berechnet:

Über Tastatur werden die zwei Elemente (müssen jeweils verschieden sein) der Menge A eingegeben. Dann werden über Tastatur die zwei Elemente (müssen jeweils verschieden sein) der Menge B eingegeben.

Wurden für eine Menge zwei gleiche Zahlen eingegeben, muß das Programm **sofort** beendet werden und eine entsprechende Meldung auf den Bildschirm ausgegeben werden.

(insbesondere darf dann nicht mehr eine weitere Zahl eingegeben und der Durchschnitt und die Vereinigung berechnet werden).

Dies muß mit dem EVA-Prinzip realisiert werden.

Außerdem muß in der Ausgabe noch angegeben werden, in welcher Menge

(z.B. 2. Menge) 2 gleiche Zahlen eingegeben wurde (und welcher Wert eingegeben wurde).

Erstellen Sie das dazugehörige C-Programm.

Bemerkungen:

a) Es widerspricht nicht dem EVA-Prinzip, wenn z.B. im Eingabeteil in bestimmten Variablen bestimmte Zustände abgespeichert werden (z.B: ob in der 1. Menge zwei gleiche Zahlen eingegeben wurden) und diese dann im Ausgabeteil abgeprüft werden.

b)

return darf nur einmal (am Ende des Programms) benutzt werden.

c) Im Ergebnis darf ein Element auch nur einmal vorkommen, also z.B: {1; 2; 3} und nicht {1; 2; 3; 2}



*Name, Vorname:*

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1) 4P

- a) Was bedeutet fußgesteuerte Schleife und wie wird diese auch noch genannt?
- b) Was bedeutet kopfgesteuerte Schleife und wie wird diese auch noch genannt?

2) 4P

Erstellen Sie das zu der folgenden Anweisung gehörige Flußdiagramm:

```
do{  
    A;  
}  
while (B);
```

3) 12P

- a) Erstellen Sie einen Programmausschnitt einer while-Schleife und einer do-while-Schleife, die jeweils den gleichen Schleifenkörper, die gleiche Schleifenbedingung und die gleichen Anweisungen vor der Schleife haben, deren Schleifenrumpf aber gleich oft durchlaufen werden
- b) Erstellen Sie einen Programmausschnitt einer while-Schleife und einer do-while-Schleife, die jeweils den gleichen Schleifenkörper, die gleiche Schleifenbedingung und die gleichen Anweisungen vor der Schleife haben, deren Schleifenrumpf aber verschieden oft durchlaufen werden

4) 3P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

```
i = 100;  
while(i!=5) {  
    i = i-2;  
    printf("%d\n", i);  
}
```

5)

4P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

```
i=10;
for(i=0;i<20;i=i+1){
    printf("Hallo Welt\n");
}
```

6)

3P

Wie viele Ausgaben auf dem Bildschirm erzeugt der folgende syntaktisch korrekte Programmausschnitt? Begründen Sie.

```
i = 100;
do{
    i = i-1;
    i = i+2;
    printf("%d\n", i);
}while (i>101);
```

7)

10P

Es sei  $a$  eine Fließkommazahl und  $n$  eine ganze Zahl.  
 $a^n$  ist wie folgt definiert:

$$a^n = \begin{cases} a * a * \dots * a & \text{(n-mal } a \text{ mit sich selbst multipliziert) , falls } n > 0 \\ 1 & \text{falls } n = 0 \\ 1 / (a * a * \dots * a) & \text{(-n-mal } a \text{ mit sich selbst multipliziert), falls } n < 0 \end{cases}$$

Implementieren Sie einen Programmausschnitt, der  $a^n$  berechnet.

Beispiele:  $a^3 = a * a * a$      $a^{-4} = 1 / (a * a * a * a)$      $a^0 = 1$

8)

10P

Das Programm (siehe unten) gibt etwas auf dem Bildschirm aus.

Die Tabelle zeigt einen 7 x 10 Bildschirm (7 Zeilen und 10 Spalten) an.

Tragen Sie die Ausgabe des Programms in die Tabelle (Bildschirm) ein.

Bem:

Schreiben Sie bei jedem Durchgang die neuen Werte der Variablen über die jeweilige Variable im Programm.


```
int main()
{
    int i,j,k;
    i=0;
    while(i<3){
        for(j=0;j<3-i;j++){
            printf(" ");
        }

        for(k=0;k<2*i+1;k++){
            printf("*");
        }
        printf("\n");

        i=i+1;
    }
    return 0;
}
```

Lösungen:

1)

4P

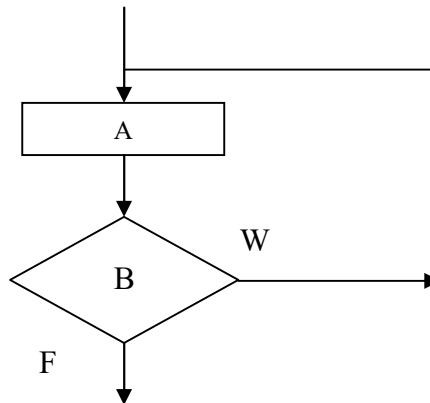
fußgesteuerte Schleife: Bedingung am Ende --> do-while-Schleife

kopfgesteuerte Schleife : Bedingung am Anfang --> while-Schleife

2)

4P

Erstellen Sie das zu der folgenden Anweisung gehörige Flußdiagramm:



3)

12P

a)

```
i=1;
while (i==1) {
    i=i+1;
}
```

```
i=1;
do{
    i=i+1;
}
while (i==1) ;
```

b)

```
while (false) {
    i = 1;
}
```

```
do{
    i = 1;
} while (false)
```

4)

3P

Unendlich viel Ausgaben

Da der Wert von i immer gerade ist, wird die Bedingung nie falsch.

5)

4P

Ein Mal, weil

```
printf("Hallo Welt\n");
```

nicht zum Körper der for-Anweisung (Semikolon beachten !) gehört.

6)

3P

Es wird eine Ausgabe auf dem Bildschirm erzeugt.

Da i nach dem 1. Durchgang den Wert 101 hat, wird die Bedingung falsch.

7)

```

...
double a;
double erg;
int n;

erg=1;

if(n>0){
    for(i=0;i<n;i++){
        erg=erg*a;
    }
}
if(n==0){
    erg=1;
}
if(n>0){
    for(i=0;i<n;i++){
        erg=erg*a;
    }
    erg=1/erg;
}

```

8)

[illegible]

## KLAUSUR 2 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vormane\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1) 50 P

Eine natürliche Zahl  $n$  ( $n \geq 2$ ) heißt Primzahl, wenn sie nur durch 1 und sich selbst teilbar ist.  
Beispiele für Primzahlen: 2, 3, 5, 7, 11, 13, ....

a) Erstellen Sie ein C-Programm, das die ersten 100 Primzahlen auf dem Bildschirm ausgibt (EVA-Prinzip muß nicht eingehalten werden).

b) Erstellen Sie dazu ein Testprotokoll mit 3 verschiedenen Tests.

## KLAUSUR 2 Programmierpraktikum 2BK11 20.1.2016 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

### AUFGABEN

1) 50P

Schreiben Sie ein C-Programm, das den Mittelwert von den vom Anwender über Tastatur eingegebenen Schulnoten berechnet und auf dem Bildschirm ausgibt.

Solange der Anwender eine Zahl eingibt, die eine Schulnote ist, wird er aufgefordert eine neue Note einzugeben.

Wenn der Anwender eine Zahl eingibt, die keine Schulnote ist, wird der Mittelwert aller bisher eingegebenen Noten berechnet und auf dem Bildschirm ausgegeben..

Bemerkung:

1)

Da hier das EVA-Prinzip nicht eingehalten werden kann, muß zumindest der Ausgabe-Teil vom EV-Teil getrennt sein. Ausgabe-Teil durch Kommentar angeben.

2)

Am Anfang des Programms muss für den Anwender eine vollständige Programminformation (Programminfo) auf dem Bildschirm ausgegeben werden, in der beschrieben steht, wie der Anwender das Programm zu bedienen hat und was das Programm macht (z.B. kann dies durch die Angabe eines Beispiels ergänzt werden).

## Lösung:

```
int main(){
    double note;
    int anzahl=0;
    double mittelwert=-1;
    double summe=0;
    int istNote=1;

    while(istNote==1){
        anzahl=anzahl+1;
        printf("Bitte Note eingeben\n");
        scanf("%lf",&note);
        if(note<1 || note >6){
            istNote=0;
        }
        else{
            summe=summe+note;
            mittelwert=summe/anzahl;
        }
    }
    if(mittelwert==-1){
        printf("es gibt keinen Mittelwert, da keine Note eingegeben wurde\n");
    }
    else{
        printf("Mittelwert=%lf\n",mittelwert);
    }

    return 0;
}
```

## KLAUSUR 2 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

### AUFGABEN

1) 50P

Schreiben Sie ein Programm, das den Ersatzwiderstand (in Ohm) von den vom Anwender über Tastatur eingegebenen, parallel geschalteten Widerständen berechnet und auf dem Bildschirm (in Ohm) ausgibt.

Solange der Anwender einen negativen Widerstand eingibt, solange wird er aufgefordert, einen positiven Widerstand einzugeben.

Gibt der Anwender den Widerstandwert 0 ein, wird das Programm beendet und der Ersatzwiderstand aller bisher eingegeben Widerstände  $> 0$  berechnet.

Bemerkung:

Für den Ersatzwiderstand  $r_{\text{Ersatz}}$  von  $n$  parallel geschalteten Widerständen  $r_1, \dots, r_n$  gilt:

$$\frac{1}{r_{\text{Ersatz}}} = \frac{1}{r_1} + \frac{1}{r_2} + \dots + \frac{1}{r_n}$$



Lösung:

```
#include "stdafx.h"

int main(int argc, char* argv[])
{
    double ersatz;
    double w;
    double summe=0;
    int beenden=0;

    do{
        printf("Bitte einen Widerstandswert in Ohm > 0  
eingeben\n");
        scanf("%lf",&w);
        if(w>0){
            summe=summe+1/w;
        }
        if(w==0){
            beenden=1;
        }

    }while(w!=0 && beenden==0);

    if(summe!=0)
        ersatz=1/summe;

    if(summe==0){
        printf("Sie haben keinen Widerstand > 0  
eingegeben\n");
    }
    else{
        printf("Ersatzwiderstand = %f\n",ersatz);
    }
    return 0;
}
```

Name, Vorname:

Hilfsmittel:  
Prioritätentabelle

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1)

12P

Betrachten Sie den folgenden Programmausschnitt:

```
...
int main() {
    int zahl = 4;
    int quad = 8;
    —————→ 1
    quadrat(zahl, &quad);
    —————→ 2
    printf("%d hoch 2 = %d\n", zahl, quad);
    return 0;
}

void quadrat(int z, int *zq) {
    *zq = z * z;
}
```

Durch die Deklaration der Variablen zahl und quad werden die folgenden Zellen

	Adresse	Inhalt
zahl	08151	?
quad	04711	?

im Arbeitsspeicher reserviert.

- a) Welchen Wert hat der Inhalt der Variablen zahl und quad an der Stelle 1 im Programm ?
- b) Welchen Wert hat z und zq beim Aufruf von quadrat(zahl, &quad) ?
- c) Was bewirkt die Anweisung \*zq = z \* z an welcher Adresse im Arbeitsspeicher ?  
(konkreten Wert der Adresse und deren Inhalt angeben!)
- d) Welchen Wert hat der Inhalt der Variablen zahl und quad an der Stelle 2 im Programm ?

2)

13P

Sie wollen ein Programm schreiben, das abhängig von der Eingabe entweder die Summe oder die Differenz oder das Produkt oder den Quotienten zweier Zahlen liefert.

Sie wollen dazu die Funktion `tr` (wie Taschenrechner) benutzen, die Sie aber wegen Arbeitsüberlastung von einem "Programmierknecht" implementieren (programmieren) lassen. Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) dieser Funktion mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)

3)

12P

Betrachten Sie den folgenden Programmausschnitt:

```
...
int main() {
    float r;
    float u;
    r = 2;
    u = 3;
    → 1
    berechne_umfang (r, u);
    → 2
    printf("Radius= %f, Umfang= %f", r, u);
}

void berechne_umfang(float radius, float umfang) {
    umfang = 2 * 3.14 * radius ;
}
```

Durch die Deklaration der Variablen `r` und `u` werden die folgenden Zellen

	Adresse	Inhalt
<code>r</code>	0120	?
<code>u</code>	0130	?

im Arbeitsspeicher reserviert.

a) Welchen Wert hat der Inhalt der Variablen `r` und `u` an der Stelle 1 im Programm ?

b) Welchen Wert hat `radius` und `umfang` beim Aufruf von `berechne_umfang (r, u)` ?

c) Was bewirkt die folgende Anweisung im Arbeitsspeicher an der Adresse 0130 ?

`umfang = 2 * 3.14 * radius;`

d) Welchen Wert hat der Inhalt der Variablen `r` und `u` an der Stelle 2 im Programm ?

4)

13P

Programmieren Sie anhand der folgenden Beschreibung die dazugehörige Funktion:

```

/*****
/**
/**  int ersatz(double r1, double r2, int mod, double *rg)  **/
/**
/**
/**#*****/
/*

```

Parameter:

- (i) double r1>0: erster Widerstandswert
- (i) double r2>0: zweiter Widerstandswert
- (i) int mod: 10: Parallelschaltung  
20: Reihenschaltung
- (o) double \*rg: Gesamtwiderstand

Return:

- (o) 0: Parallelschaltung oder Reihenschaltung wurde  
berechnet (mod ist 10 oder 20)
- 1: mod ist weder 10 noch 20

Beschreibung:

Berechnet in Abhängigkeit vom Modus mod (10 bedeutet eine Parallelschaltung, 20 bedeutet eine Reihenschaltung), den Widerstandswerten r1 und r2 den Ersatzwiderstand (Gesamtwiderstand) rg der Widerstandsschaltung.

\*/

## Lösungen

- 1) 12P  
a) zahl: 4, quad: 8 1P + 1P  
b) z: 4, zq: 04711 1P + 2P  
c) In den Inhalt der Adresse 04711 wird der Wert 16 geschrieben. 4P  
d) zahl: 4, quad: 16 1P + 2P

2) 13P  
/\*\*\*\*\*/  
/\*\* \*\*/  
/\*\* double tr (double z1, double z2, int mod) \*\*/  
/\*\* \*\*/  
/\*#\*\*\*\*\*/

Parameter:

- (i) double z1: erste Zahl
- (i) double z2!=0, wenn mod=4: zweite Zahl
- (i) int mode{1;2;3;4}:
  - 1: berechnet Summe z1+z2
  - 2: berechnet Differenz z1-z2
  - 3: berechnet Produkt z1\*z2
  - 4: berechnet Quotient z1/z2

Return:

- (o) Ergebnis der gewünschten Operation (Summe, oder Differenz oder Produkt oder Quotient).

Beschreibung:

Berechnet in Abhängigkeit vom Modus mod:

Summe z1+z2, wenn mod = 1

Differenz z1-z2, wenn mod = 2

Produkt z1\*z2, wenn mod = 3

Quotient z1/z2, wenn mod = 4

\*/

Jede fehlende Zusicherung: -2 P

z2!=0 ist keine richtige Zusicherung, da dann z.B. auch 3 \* 0 verboten würde: -2 P

Bemerkung zum Begriff Zusicherung:

Die Angabe beim Parameter z1:

z2!=0, wenn mod=4

und die Angabe:

mode{1;2;3;4}

nennt man Zusicherung. Das bedeutet, daß unter diesen Voraussetzungen der Programmierer dieser Funktion für die Korrektheit der Berechnungen dieser Funktion **garantiert**.

Je weniger Zusicherungen der Programmierer macht, desto größer wird der programmtechnische Aufwand für ihn, desto mehr "Intelligenz" muss er in die Funktion packen.

- 3) 12P  
a)  $r = 2, u = 3$  1P + 1P  
b) radius = 2, umfang = 3 1P + 2P  
c) nichts 4P  
d)  $r = 2, u = 3$  3P

4) 13P

```
int ersatz(double r1, double r2, int mod, double *rg){
    int r;
    if(mod==10){
        *rg=1/(1/r1+1/r2);
        r=0;
    }
    else if(mod==20){
        *rg=r1+r2;
        r=0;
    }
    else
        r=-1;
    return(r);
}
```

# KLAUSUR 3 Programmierpraktikum 2BK11 27.4.2016 Zeit: 60 Minuten

Name, Vorname:

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

In einem aus Zeichen bestehenden Eingangsfeld (Array) sollen zwischen 2 Stellen (jeweils einem Index) Zellen mit einem bestimmten Zeichen überschrieben werden. Diese Veränderungen dürfen aber nicht im Eingangsfeld gemacht werden. Wenn sich eine der 2 Stellen außerhalb des Feldes befinden, muß darauf entsprechend reagiert werden (entsprechender Output, keine Fehlermeldung auf dem Bildschirm)

Beispiel: In der Zeichenfolge "Mathematik" alle Zeichen zwischen der Stelle 4 und 7 mit dem Zeichen 'x' überschreiben.

Eingangsfeld (Array)	Ausgangsfeld (Array)
Mathematik	Mathxxxxik

a)

20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion `overwrite(...)` nach dem im Unterricht verwendeten Schema. Wo nötig (bzw. von Vorteil) den Bezeichner `const` verwenden.

b)

20P

Implementieren Sie die Funktion " `overwrite(...)` "

c)

10P

In `main` muß ein Aufruf der Funktion " `overwrite(...)` " realisiert werden.  
(keine Eingabe über `scanf()`, sondern Aufruf mit konkreten Parametern).

Testen Sie diese Funktion, indem die Elemente des Ausgangsfeldes (Array) auf dem Bildschirm ausgegeben werden.

## Lösung:

```
#include "stdafx.h"
```

```
int overwrite(const char stringIn[], char stringOut[], int begin,  
             int end, char zeichen);
```

```
int main(int argc, char* argv[]){  
    int erg;  
    int i;  
    char stringIn[]="abcdefg";  
    char stringOut[10];  
  
    erg=overwrite(stringIn, stringOut, 0, 6, 'x');  
  
    i=0;  
    if(erg!=0)  
        return 0;  
  
    while(stringOut[i]!='\0'){  
        printf("%c",stringOut[i]);  
        i++;  
    }  
  
    printf("\n\n");  
  
    return 0;  
}
```

```
/*  
**  
** int overwrite(const char stringIn[], char stringOut[], int begin,  
**             int end, char zeichen)  
**  
**  
*#  
*/
```

### Parameter:

- (i) const char stringIn[]: Eingangsfeld
- (o) char stringOut[] : Ausgangsfeld
- (i) int begin<=end : linke Stelle
- (i) int end : rechte Stelle
- (i) char zeichen : Zeichen mit dem überschrieben wird

### Return:

- 1 : begin>=len oder begin<0 oder end>=len oder end<0  
oder begin>end  
wobei len=Länge von stringIn
- 0 : sonst

### Beschreibung:

Wenn sich "end" und "begin" innerhalb von stringIn befinden, werden von der Stelle "begin" (Zählung beginnt bei 0) bis zur Stelle "end" alle Zeichen des Strings "stringIn" mit dem Zeichen "zeichen" überschrieben und 0 zurückgeliefert, sonst wird dies nicht gemacht und -1 zurückgeliefert. Diese Aktion wird aber nicht in stringIn gemacht, sondern das Ergebnis nach stringOut kopiert.

```
*/
```



```
int overwrite(const char stringIn[], char stringOut[], int begin,
              int end, char zeichen){
    int i=0;
    int len;

    while(stringIn[i]!='\0'){
        stringOut[i]=stringIn[i];
        i++;
    }
    stringOut[i]='\0';
    len=i;

    if(begin>=len || begin<0 || end>=len || end<0)
        return(-1);

    for(i=begin; i<=end; i++){
        stringOut[i]=zeichen;
    }
    return 0;
}
```

# KLAUSUR 3 Programmiertheorie 2BK11 Nachtermin 1 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:

Schema einer Dokumentation einer Funktion

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1)

Die Funktion atoi(...) berechnet die zu einer nur aus Ziffern bestehenden Zeichenfolge den Wert der zugehörigen Zahl.

Beispiel:

Zeichenfolge: "345"

Zugehörige Zahl:  $3 \cdot 100 + 4 \cdot 10 + 5 \cdot 1$

Es darf vorausgesetzt werden, daß die Zeichenfolge nur aus Ziffern besteht und leer ist.

Sie sollen dazu eine möglichst "luxuriös" gestaltete **Funktion** benutzen, die Sie aber wegen Arbeitsüberlastung im Fach Mathematik von einem "Programmierknecht" implementieren (programmieren) lassen.

Bemerkung: Ausschnitt aus der ASCII-Tabelle:

'0' --> 48, '1' --> 49, '2' --> 50, usw.

a)

15P

Entwerfen Sie für den "Programmierknecht" eine **Beschreibung** (Leistungsbeschreibung) der Funktion "atoi" mit dem im Unterricht verwendeten Schema. (Kein Programm !!!)

Überlegen Sie sich zuerst genau, welche **Parameter** nötig sind

b)

30P

Da sich der Programmierknecht zur Zeit in einem sogenannten "Tschill-Urlaub" befindet und deshalb aktuell (und wohl auch zukünftig) nicht ansprechbar ist, muss die Funktion "atoi" von Ihnen selbst implementiert werden. Implementieren Sie die Funktion "atoi".

c)

5P

Schreiben Sie ein Programm mit einem Aufruf der Funktion "atoi"

(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

## Lösungen:

1a) 15P  
/\*\*\*\*\*  
/\*\*  
/\*\* void atoi(char str[]) \*\*/  
/\*\* \*\*/  
/\*#\*\*\*\*\*  
/\*

Parameter:

- (i) char str[]!='\0' : String  
und alle Zellen  
des Feldes Ziffern

Return:

- (o) die als Zahl interpretierte Zeichenfolge str

Beschreibung:

berechnet die zu einer nur aus Ziffern bestehenden  
Zeichenfolge den zugehörigen Zahlenwert.

Beispiel:

```
erg : ?  
erg=atoi("58");  
erg : 58  
*/
```

b) 30P

```
int atoi(char str[]){  
    int i=0;  
    int len;  
    int sum=0;  
    int faktor = 1;  
  
    for(i=0;str[i]!='\0';i++){  
    }  
    len=i;  
  
    for(i=len-1;i>=0;i--){  
        sum = sum + (str[i]-48)*faktor;  
        faktor = faktor * 10;  
    }  
  
    return sum;  
}
```

c) 5P

```
erg=atoi("345");
```

# KLAUSUR 3 Programmierpraxis 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programnteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programnteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

Die Funktion split2(...) teilt eine Zeichenfolge in 2 Teilzeichenfolgen, wobei das erstmalige Vorkommen eines bestimmten Zeichens die erste Teilzeichenfolge begrenzt.

Beispiel:

Zeichenfolge: "abcasdfgfdgdf"

Begrenzer: ' f '

Die zwei Teilzeichenfolgen:

"abcasd" und "gfdgdf"

Sie sollen dazu eine möglichst "luxuriös" gestaltete **Funktion** benutzen, die Sie aber wegen Arbeitsüberlastung im Fach Mathematik von einem "Programmierknecht" implementieren (programmieren) lassen.

a) 20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion split2(...) nach dem im Unterricht verwendeten Schema. Wo nötig (bzw. von Vorteil) den Bezeichner const verwenden.

b) 20P

Implementieren Sie die Funktion split2(...)

c) 10P

In main muß ein Aufruf der Funktion split2(...) realisiert werden.

(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

Testen Sie diese Funktion, indem die Elemente der zwei gesplitteten Zeichenfolgen auf dem Bildschirm ausgegeben werden.

## Lösungen:

1a)

```
/* **** */
/**
/** void split2(char str1[], char begrenzer, char str2[]) **/
/**
/*# **** */
/*
```

Parameter:

- (i/o) char str1[]: String, der in 2 Strings aufgeteilt werden soll. Außerdem wird dort auch der 1. String links des Begrenzers abgespeichert (als output)
- (i) char begrenzer: Zeichen, des 1. Vorkommen die Zerlegung angibt.
- (i) char str2[] : Der String rechts des Begrenzers.

Return:

kein

Beschreibung:

Zerlegt den String str1 an der Stelle des Begrenzers (erstmaliges Vorkommen) in die 2 Teilstrings str1 und str2  
Kommt der Begrenzer nicht in str1 vor, bleibt str1 unverändert und str2 = '\0'

Beispiel:

```
str1[]: "abcfxyzffz"
begrenzer: 'f'
str2[]: "wwwwwwwwwwww"
split2(str1, begrenzer, str2);
str1[]: "abc"
begrenzer: 'f'
str2[]: "xyzffz"
*/
```

b)

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

```
void split2(char str1[], char begrenzer, char str2[]);
```

```
int main(){
    char str1[]="abdexgijkf";
    char str2[]="xxxxxxxxxxxxxxxxxx";
    char begrenzer = 'f';
    printf("str1= %s\n",str1);
    split2(str1, begrenzer, str2);
    printf("str1= %s\n",str1);
    printf("str2= %s\n",str2);

    return 0;
}
```

```

c)
void split2(char str1[], char begrenzer, char str2[]){
    int len;
    int i;
    int index;

    index=-1;
    len=0;

    // Länge der zu splittenden Zeichenkette bestimmen
    for(i=0;str1[i]!='\0';i++){
        len++;
    }

    // Index des Begrenzers bestimmen
    for(i=0; str1[i]!='\0'; i++){
        if(str1[i]==begrenzer){
            index=i;
            break;
        }
    }

    if(index==-1)
        str2[0]='\0';
    else{
        str1[index]='\0';
        for(i=0;str1[index+i+1]!='\0';i++){
            str2[i]=str1[index+1+i];
        }
        str2[i]='\0';
    }
}

```

# KLAUSUR 4 Programmierpraktikum 2BK11 22.6.2016 Zeit: 45 Minuten

Name, Vorname:

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

In einem aus Zeichen bestehenden Eingangsfeld (Array) soll jedes Auftreten eines bestimmten Zeichens gelöscht werden.

Diese Veränderungen dürfen aber nicht im Eingangsfeld gemacht werden.

Beispiel: In der Zeichenfolge "Mathematik" jedes Zeichen 'a' löschen.

Eingangsfeld (Array)	Ausgangsfeld (Array)
Mathematik	Mthemtik

a)

20P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion entferneZeichen(...) nach dem im Unterricht verwendeten Schema.

b)

20P

Implementieren Sie die Funktion " entferneZeichen(...)"

c)

10P

In main muß ein Aufruf der Funktion "entferneZeichen(...)" realisiert werden.

(keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern).

Testen Sie diese Funktion, indem die Elemente des Ausgangsfeldes (Array) auf dem Bildschirm ausgegeben werden.

Beispiel einer Dokumentation einer Funktion:

```

/*****
**
**  int berechne(double a, double b, double *u, double *f)  **
**
**#*****
*/
Parameter:
    (i) double a>=0 :   eine Seitenlänge des Rechtecks
    (i) double b>=0 :   andere Seitenlänge des Rechtecks
    (o) double *u    :   Umfang des Rechtecks
    (o) double *f    :   Fläche des Rechtecks

Return:
    (o) 1:  Quadrat
        2:  Rechteck

Beschreibung:
    Berechnet aus den beiden positiven Seitenlängen a und b des Rechtecks
    den Umfang *u und die Fläche *f und bestimmt außerdem, ob es
    ein Quadrat oder ein Rechteck ist.

Beispiel:
    a:10 , b:20
    erg=berechne(a, b, &umfang, &flaeche)
    umfang:60 , flaeche:200 , erg:2
*/

int berechne(double a, double b, double *u, double *f){
    int r;
    *u=2*(a+b);
    *f=a*b;
    if(a==b){
        r=1;
    }
    else{
        r=2;
    }
    return(r);
}

```



## Lösung:

```
#include "stdafx.h"
```

```
void entferneZeichen(char stringIn[], char c, char stringOut[])
```

```
int main()
{
    int z;
    char stringIn[]="abcxyzx";
    char stringOut[]="-----";
    entferneZeichen(stringIn, 'x', stringOut);
    printf("%s",stringOut);
    scanf("%d",&z);
    return 0;
}
```

```

/*****
**
** void entferneZeichen(char stringIn[], char c, char stringOut[]) **
**
**#*****/
/*
```

Parameter:

- (i) char stringIn[]: Eingangsfeld
- (i) char c : zu entfernendes Zeichen
- (o) char stringOut[] : Ausgangsfeld

Return:

nichts

Beschreibung:

Alle Zeichen "c", die sich in "stringIn" befinden, werden gelöscht.

Diese Aktion wird aber nicht in "stringIn" gemacht, sondern das Ergebnis nach "stringOut" kopiert.

```
*/
void entferneZeichen(char stringIn[], char c, char stringOut[]){
    int anz;
    int i,j;
    int len;
    int erg;
    erg=0;

    i=0;
    j=0;
    for(i=0; stringIn[i]!='\0'; i++){
        if(stringIn[i]!=c){
            stringOut[j]=stringIn[i];
            j++;
        }
    }
    stringOut[j]='\0';
    return;
}
```

*Name, Vorname:*

Hilfsmittel:  
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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

## AUFGABEN

1) 13P

Im Fach Mathematik soll eine "Polynomfunktion 2. Grades" (" $y = ax^2 + bx + c$ ") erstellt werden. Der Mathelehrer gibt diese Aufgabe an die EDV-Abteilung weiter.

a) 5P

Machen Sie zuerst eine Leistungsbeschreibung (Dokumentation) dieser Funktion fParabel(...) nach dem im Unterricht verwendeten Schema.

b) 4P

Implementieren Sie die Funktion fParabel(...)

c) 4P

Es soll der y-Wert einer um in y-Richtung um 2 nach oben verschobenen Normalparabel an der x-Stelle 4 berechnet werden.

Schreiben Sie ein Programm, in dem ein dazu entsprechender Aufruf der Funktion fParabel(...) verwendet wird.

Keine Eingabe über scanf() !!

2) 13P

Die Funktion kehrwerte(...) bildet die Kehrwerte eines i/o-Feldes der Länge 2 von double-Werten in aufsteigender Reihenfolge. Diese Kehrwerte müssen also im gleichen Feld gespeichert werden.

Beispiel: 3, 4 --> 1/4 , 1/3

a) 10P

Implementieren Sie diese Funktion kehrwerte(...).

b) 3P

Verwenden Sie einen Aufruf dieser Funktion, um die Kehrwerte des Feldes zu bilden, in dem die Zahlen 8 und 7 gespeichert sind.

Keine Eingabe über scanf() !!



## Lösungen:

1)

a) 5P

```
/******  
/**  
/** double fParabel(double a, double b, double c, double x) **/  
/** **/  
/*#*****  
/*
```

Parameter:

- (i) double a : Parameter a der Parabel
- (i) double b : Parameter b der Parabel
- (i) double c : Parameter c der Parabel
- (i) double x-Wert der Parabel

Return:

$ax^2+bx+c$

Beschreibung:

Berechnet den y-Wert einer Parabel mit der Funktionsgleichung  $y=ax^2+bx+c$  an der Stelle x.

\*/

b) 4P

```
double fParabel(double a, double b, double c, double x){  
    return (a*x*x+b*x+c);  
}
```

c) 4P

```
int main(){  
    double y;  
    y= fParabel(1,0,2,4);  
    printf("%f",y);  
    return 0;  
}
```

2)

a) 10P

```
void kehrwerte(double v[]){  
    double temp0, temp1;  
    temp0=1/v[0];  
    temp1=1/v[1];  
  
    if(temp0<temp1){  
        v[0]=temp0;  
        v[1]=temp1;  
    }  
    else{  
        v[0]=temp1;  
        v[1]=temp0;  
    }  
}
```

b) 3P

```
int main(){  
    int feld[]={7,8};  
    kehrwerte(feld);  
    return 0;  
}
```

3)

13P

a) 6P

```
void anfüegen(double feld[], double element){
    int len;
    len = feld[0];
    feld[len+1] = element;
    feld[0]++;
    return;
}
```

b) 4P

```
int main(){
    double feld[1000];
    feld[0]=1;
    anfüegen(feld, 19);
    anfüegen(feld, 13);
    return 0;
}
```

c) 3P

weil dann die Anzahl der anzufügenden Elemente beliebig groß sein darf und nicht durch die beim Deklarieren angegeben Länge des Feldes beschränkt wird.

4)

a)

4P

```
void belegen(int p[], int anz, int wert){
    int i;
    for(i=0; i<anz; i++){
        p[i]=wert+1;
    }
}
```

b)

4P

7	7	7	7	7	7	7	7	7	7
---	---	---	---	---	---	---	---	---	---

c)

5P

7	7	3	3	7	7	7	7	7	7
---	---	---	---	---	---	---	---	---	---

# KLAUSUR 4 Programmiertheorie 2BKI1 Nachtermin 1 Zeit: 90 Minuten

Name, Vorname:

Hilfsmittel:

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.
- return genau einmal am Ende des Quellcodes einer jeder Funktion (u.a. auch main() ) benutzen.
- Programme müssen benutzerfreundlich sein.
- EVA-Prinzip muss benutzt werden.
- Einrücken der entsprechenden Programmteile.
- Bei Nichtbeachtung dieser Hinweise gibt es einen Punkteabzug !!!!  
Bei praktischen Arbeiten am Computer bitte folgendes beachten:
- NUR **ablauffähige** C-Programme werden bewertet ! **Jede** Aufgabe als eigenes Projekt (keine Warnungen vom Compiler).
- Der Name, Vorname und die Aufgabennummer (z.B: Aufgabe 1) muß als **Kommentar** in jedes Programm mit aufgenommen werden.
- Name der Projekte nach folgendem Schema vergeben: Gruppe\_Rechnernummer\_Nachname\_Vorname\_Aufgabennr.  
Beispiel: A\_12\_Mustermann\_Erika\_nr3
- Die Programme müssen ausgedruckt werden.
- Jeder Programmteil (Eingabe, Verarbeitung, Ausgabe) muss als solcher durch einen Kommentar angegeben werden.
- Jedes Projekt (d.h. jeder Projektordner) muß auf den Lehrerstick kopiert werden.

Bemerkungen:

Der Compiler darf keine Warnungen und Fehler erzeugen !!!

## AUFGABEN

1)

a)

40P

Eines morgens in aller Frühe finden Sie auf Ihrem Schreibtisch die folgende Leistungsbeschreibung (Dokumentation) der Funktion "ersetzen (...)" vor (siehe Rückseite). Implementieren Sie diese Funktion.  
Wo nötig (bzw. von Vorteil) den Bezeichner const verwenden.

b)

5P

Schreiben Sie ein Programm, in dem ein Aufruf der Funktion "ersetzen (...)" verwendet wird (keine Eingabe über scanf(), sondern Aufruf mit konkreten Parametern). Testen Sie diese Funktion, indem die Elemente der neuen Folge auf dem Bildschirm ausgegeben werden.

c)

5P

Welchen Nachteil hätte es, wenn man in der Funktion ersetzen(...) dynamisch Speicherplatz für "neueFolge" belegt?  
Man könnte ja die Größe dieses Speicherplatzes innerhalb der Funktion ersetzen(...) berechnen lassen und dann genau so viel Speicherplatz für "neueFolge" allokieren, wie "neueFolge" benötigt.

```

/*****
/**
/** void ersetzen(char zeichen, char folge[],
/** char ersetzung[], char neueFolge[])
/**
/**
/*****
/*

```

Parameter:

- (i) char zeichen : zu ersetzendes Zeichen.
- (i) char folge[] : dort wird das "zeichen" ersetzt.
- (i) char ersetzung[]: Das Zeichen "zeichen" wird durch die Zeichenfolge "ersetzung" ersetzt.
- (o) char neueFolge[] : Die durch die Ersetzung entstehende neue Zeichenfolge

Return:

kein

Beschreibung:

In der Zeichenfolge "folge" wird beim erstmaligen Auftauchen des Zeichens "zeichen" dieses Zeichen durch die Zeichenfolge "ersetzung" ersetzt. Dadurch entsteht die neue Zeichenfolge "neueFolge".

Der Programmierer, der diese Funktion benutzt muß dafür Sorge tragen, dass beim Aufruf dieser Funktion genügend Speicherplatz für "neueFolge" bereitgestellt wird.

Beispiel 1:

zeichen: a  
folge: "raav"  
ersetzung: "xy"  
Nach dem Aufruf:  
neueFolge: "rxyav"

Beispiel 2:

zeichen: s  
folge: "raav"  
ersetzung: "xy"  
Nach dem Aufruf:  
neueFolge: "raav"

\*/

## Lösungen:

```
#include "stdafx.h"
#include <string.h>
#include <malloc.h>

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]);

int main(int argc, char* argv[]){
    char folge[4]="raf";
    char ersetzung[3]="xy";
    char neueFolge[4];
    char zeichen='a';

    ersetzen(zeichen, folge, ersetzung, neueFolge);
    printf("neueFolge=%s\n",neueFolge);
    return 0;
}

void ersetzen(const char zeichen, const char folge[],
              const char ersetzung[], char neueFolge[]){
    int i=0;
    int j=0;
    int index=0;

    // Kopiere alle Elemente von folge bis zum Auftreten
    // des gefundenen Zeichens in neueFolge
    while(folge[i]!='\0' && folge[i]!=zeichen){
        neueFolge[i]=folge[i];
        i++;
    }
    // Zeichen wurde nicht gefunden
    if(i==strlen(folge)){
    }
    else{
        index=i;
        // Füge ersetzung an neueFolge an
        while(ersetzung[j]!='\0'){
            neueFolge[i]=ersetzung[j];
            i++;
            j++;
        }
        j = index+1;
        // Füge von j=index an die folge an die neue Folge an.
        while(folge[j]!='\0'){
            neueFolge[i]=folge[j];
            i++;
            j++;
        }
    }
    neueFolge[i]='\0';
}
```