Beispiel 1

(3 Punkte)

```
//
// programme:  simpleInput2.c
// author:     Josef Strasser-Leitner
// date:       01.10.05
// version:    1.00
// purpose:    programming course C - simple input and if-statement


#include <stdio.h>;

/main programme
// parameter values: none
// return value: 0 or 1
int main()
{
  int age=0;
  printf("Wie alt bist Du: ")
  scanf("%d",&age);
  if ( ( age > 0 ) AND ( age < 100 ) ) {
    printf("Ich wiederhole: " %d " Jahre.\n", age);
    return 0;
  } // end if
  else {
    printf("Fehler in der Eingabe!\n";
    return 1;
  } // end else

  return 0; // this line will not be executed because of the
            // returns before
} // end main
```

Geben Sie dieses Programm ein und kompilieren Sie es. Allerdings haben sich im Quelltext ein paar Fehler eingeschlichen! Finden Sie die Fehler.

Beispiel 2
(3 Punkte)

Ein- und Ausgabe von Zahlen (u.a. mit einer Kommastellen) mit `scanf()` und `printf()` und Division von Zahlen

Lesen Sie über Tastatur das Alter (in Jahren) Ihrer beiden Nachbarn und Ihr eigenes Alter ein. Geben Sie anschließend die eingelesenen Altersangaben (als Ganzzahl) und das errechnete Durchschnittsalter (mit einer Kommastelle) wieder aus.

Speichern Sie das Programm mit dem Namen `average-age.c`.

Beispiel 3
(5 Punkte)

Konstanten und if-Anweisunge
Erstellen Sie ein Programm, dass wahlweise Kreis- und Kugelberechnungen durchführt.
Der Programmlauf soll wie folgt sein: Lesen Sie den Radius von Tastatur ein. Das Programm soll anschließend fragen, ob Kreis- oder Kugelberechnungen durchgeführt werden sollen. Lösen Sie das Problem mit einer `if`-Struktur.
Bei Kreisberechnungen sollen der Umfang und die Fläche, bei Kugelberechnungen Umfang, Oberfläche und Volumen berechnet und ausgegeben werden.

Definieren Sie **pi** als Konstante einmal mit `const` und einmal als Makro mit `define`:

```
const double PI = 3.14159265358979323846;
```
Im ‚Deklarationsbereich' der `main()`-Funktion.

```
#define PI 3.14159265358979323846 (ohne ; am Zeilenende!)
```
Nach der Zeile `#include`. Sogenannte Präprozessor-Anweisungen werden außerhalb der `main()`-Funktion im ‚globalen' Bereich am Anfang des Programmes eingetragen.

Bezeichner für Konstanten werden üblicherweise immer groß geschrieben. Beachten Sie auch, dass Fließkommazahlen mit Punkt anstatt Komma eingegeben werden.

Speichern Sie die beiden Programme mit `circle-ball1.c` und `circle-ball2.c`.

Beispiel 4
(2 Punkte)

Erstelle ein kleines Programm, das zwei Integer-Zahlen und zwei Real-Zahlen (Kommazahlen – Typ double oder float) von Tastatur einliest (Achtung: "Kommazahlen" werden mit Punkt eingegeben, nicht mit Komma!).

Diese beiden Werte sollen jeweils miteinander dividiert und die Ergebnisse (direkt) am Bildschirm ausgegeben werden.

Wie schaut das Ergebnis der Division zweier Integer-Zahlen aus (z.B. 5/3 oder 12/8)?

Beispiel 5
(1 Punkt)

Das folgende Programm gibt das Minimum zweier via Tastatur eingegebener Zahlen aus.

Beide Werte werden einer Funktion `minimum` übergeben, die mittels `if`-Anweisung die kleinere der beiden Zahlen zurückliefert, die dann im Hauptprogramm ausgegeben wird. Kompiliere das Programm.

```
// programme: minimum.c
// author: Josef Strasser-Leitner
// date: 02.10.2005
// version: 1.00
// purpose: function call, if-statement – programming course C

#include <stdio.h>

// function to calculate the minimum of two numbers
// parameters: int number1, number2
// return value: int -  the smaller number of the two numbers
int minimum (int number1, int number2);


// main function
// parameters: none
// return value: 0
int main()
{
  int x=0;
  int y=0;
  printf("Dieses Programm berechnet das Minimum zweier Zahlen. \n");
  printf("Gib die erste Zahl ein: ");
  scanf("%d",&x);
  printf("Gib die zweite Zahl ein: ");
  scanf("%d",&y);
  printf("Das Minimum von %d und %d ist ", x, y);
  printf("%d.\n", minimum (x, y) );
  return 0;
 } // end main

// function to calculate the minimum of two numbers
// parameters: int number1, number2
// return value: int -  the smaller number of the two numbers
int minimum (int number1, int number2)
{
  if ( number1 < number2 )
    return number1;
  else
    return number2;
}
```

Beispiel 6
(3 Punkte)

Schreib ein kleines Programm, das berechnet, ob eine eingegebene Jahrezahl ein Schaltjahr ist oder nicht.

Ein Jahr ist ein dann Schaltjahr, wenn die Jahreszahl durch 4, aber nicht durch 100 teilbar ist.
Ist das Jahr durch 400 teilbar, handelt es sich ebenfalls um ein Schaltjahr.

Das gilt allerdings nur für den gregorianischen Kalender, also seit 1582. Nach der julianischen Regel ist jedes vierte Jahr ein Schaltjahr. Anders formuliert noch einmal die gregorianischen Regel: jedes vierte Jahr ist nur dann ein Schaltjahr, wenn es nicht ohne Rest durch hundert teilbar ist, es sei denn, die Jahreszahl ist durch vierhundert ohne Rest teilbar.

Erstelle eine eigene Funktion, der die Jahrszahl übergeben wird, und die als Ergebnis `true` oder `false` zurückliefert.

Gib dann im Hauptprogramm aus, ob die eingegebene Jahreszahl ein Schaltjahr ist oder nicht.

Beispiel 7
(Kompilierung 1 Punkt)

```
// programme: change.c
// author: Josef Strasser-Leitner
// date: 02.10.2005
// version: 1.00
// Zweck: demonstration of ’call by value’ –
// programming course c
//

#include <stdio.h>


// function to change (swap) two numbers
// parameter values: int number1, int number2
// return value: none
void change (int number1, int number2)
{
  int temp = number1; //temporary auxiliary variable
  number1 = number2;
  number2 = temp;
}

// main function
// parameters: none
// return value: 0
int main()
{
  int a = 13;
  int b = 5;
  printf("%d %d\n", a , b);
  change ( a , b );
  printf("%d %d\n", a , b);
  return 0;
} // end main
```

Die Ausgabe des Programms ergibt
13 5
13 5
Warum?

Beispiel 8 – Globale Variable
(Kompilierung 1 Punkt)

```
// programme: globVar.c
// author: Josef Strasser-Leitner
// date: 02.10.2005
// version: 1.00
// purpose: demonstration global variable

#include <stdio.h>

int globVar = 1; // global variable

// function to change a global variable
// paramter values: none
// return value: none
void func ()
{
  globVar = 33;
}


// main function
// parameter values: none
// return value: 0
int main()
{
  printf("Dieses kleine Programm veraendert eine globale Variable ");
  printf("in einer Funktion.\n");
  printf("Die globale Variable vor der Veraenderung: %d\n", globVar);
                                          //output = 1
  func();
  printf("Die globale Variable nach der Veraenderung: %d\n",globVar);
                                          //output = 33
  return 0;
} // end main
```

Beispiel 9
(1 Punkt)

Verändere Beispiel 7 – `change.c` so, dass das Programm korrekt arbeitet. Verwende dabei globale Variablen wie im vorigen Beispiel 8.

Anmerkung: Die Lösung mit globalen Variablen ist eine mögliche aber nicht die beste bzw. eleganteste. Wir werden später eine elegantere Lösung mit Referenzen kennen lernen.

Speichere das Programm mit dem Namen `change2.c`.

Beispiel 10
(3 Punkte)

Berechne die Höhe eines Haus, wenn du die Distanz zum Haus kennst und den Winkel, den die Bodenfläche und die Dachspitze von deinem Standpunkt aus bilden.

Für die, die sich mit Winkelfunktionen schwer tun, die Formel:

```
height = distance *  tan(angle)
```

Um in C trigonometrische Funktionen wie sin und cos verwenden zu können, muss die Datei `<math.h>` mit `#include` eingebunden werden!

Zu beachten ist, dass in trigonom. Funktionen nur Werte in Bogenmaß (Radiant) eingegeben werden können.
Umrechnung:

```
Winkel (Radiant) = Winkel (Grad) * 2 * PI / 360
```

Deklarieren Sie PI wieder als Konstante.
Speichern Sie das Programm mit dem Namen `height.c`.

Anmerkung: Compiler-abhängig muss unter Umständen beim Kompilieren mit dem Schalter „`-lm`“ `math` dazugelinkt werden.