Beispiel 1

(3 Punkte)

Erstelle ein kleines Programm, das Grad Celsius in Grad Fahrenheit umrechnet. Du liest von der Tastatur einen Grad-Celsius-Wert ein und übergibst ihn der Funktion `celsToFahr().` Die Funktion liefert Grad Fahrenheit zurück. Der Wert wird dann am Bildschirm ausgegeben. Speichere das Programm unter `celsFahr.c`.

Beispiel 2

(2 Punkte)

Erstelle eine zweite Funktion, die Grad Fahrenheit in Grad Celsius umrechnet. Erstelle in der `main()-Funktion` ein kleines Auswahlmenü. Passe auch die Ausgabe entsprechend an.

Die Formeln:

```
celsius = (fahrenheit – 32.0 ) * 5.0 / 9.0
```

und

```
fahrenheit = 9 * celsius / 5.0 + 32
```

Achtung: Man muss den Ausdruck `5/9` als `5.0/9.0` eingeben, da sonst mit Integer-Wert gerechnet wird, und als Ergebnis von 5 / 9 der Wert 0 rauskommt, vor allem dann, wenn die 5/9 vor dem Klammerausdruck stehen: `5 / 9 * ( fahrenheit – 32)` liefert als Ergebnis immer 0!

Beispiel 3

(zur Demonstration)

```
//programme: fac.c
//author: Josef Strasser-Leitner
//date: 02.10.2005
//version: 1.00
//purpose: recursive factorial calculation – demonstration recursion

#include <stdio.h>

// purpose: to calculate the factorial of a number
// parameter values: int a (=input number)
// return value: long double (=factorial)
long double fac ( int a )
{
  if ( a > 1 )
    return a * fac( a - 1);
  else
    return 1;
}

// main function
// parameter values: none
// return value: 0
int main()
{
  long double result = 0.0;   // long double has a bigger range of
                              // values than int
  int number = 0;

  printf("Dieses Programm berechnet die Fakultaet einer Zahl. ");
  printf("Gib eine Zahl ein: ");
  scanf("%d", &number);

  result = fac( number );

  printf("\n%d! = %.0Lf\n", number, result);
  return 0;
}
```

Beispiel 4
(4 Punkte)

Versuche die Fibonacci-Zahlen rekursiv zu berechnen:
`0-1-1-2-3-5-8-13-21-34-55-89-144-usw.`

Ein Folgeglied errechnet sich dabei immer aus der Summe der beiden vorhergehenden Glieder.

Die Funktion zur Berechnung des n-ten Gliedes muss
`0` zurückliefern , wenn `n=0`, und muss
`1` zurückliefern, wenn `n=1`,
ansonsten
gib `( fibonacci(n-1) + fibonacci(n-2) )` zurück.

`n` soll von der Tastatur eingelesen werden.

Die **Fibonacci-Folge** ist eine mathematische Folge von positiven ganzen Zahlen, den **Fibonacci-Zahlen**. Der Mathematiker Leonardo Fibonacci (Leonardo von Pisa) entwickelte sie, um das Wachstum einer Population von Kaninchen zu beschreiben, und publizierte sie in seinem Buch "Liber Abaci" aus dem Jahre 1202.

Speichere das Programm unter dem Namen `fibonacci.c`.

Beispiel 5 – Switch-Anweisung

(zur Demonstration)

```
// programm: inch-cm-switch.c
// author:   Bjarne Stroustrup, modified by Josef Strasser-Leitner
// date:     02.10.2005
// version:  1.00
// purpose:  switch demonstration

#include <stdio.h>

// main function
// parameters: none
// return values: 0
int main()
{
  const float coefficient = 2.54; // 1 inch is equivalent to 2.54 cm
  float x, in, cm;
  char ch = 0;

  printf("Dieses Programm rechnet Inch in Zentimeter");
  printf(" um und umgekehrt.\n\n");
  printf("Bitte Laenge eingeben: ");
  do{ scanf("%f", &x) ; } while (getchar() != '\n');
  printf("Bitte Einheit eingeben (i oder c): ");
  do{ scanf("%c", &ch);} while (getchar() != '\n');


  //handle the input
  switch (ch)
  {
    case 'i':           // inch
         in = x;
         cm = x * coefficient;
         break;
    case 'c':           // cm
         in = x / coefficient;
         cm = x;
         break;
    default:            //optional
         in = cm = 0;
         break;
  } // end switch

  printf("%.2f in = %.2f cm\n", in, cm);

  return 0;
} // end main
```

Beispiel 6

(3 Punkte)

Erzeuge ein neues Programm, das von der Tastatur ein beliebiges Zeichen einliest und prüft, ob das eingegebene Zeichen ein Zahl zwischen 1 und 5 ist. Liegt die Zahl in diesem Intervall, ist sie auszugeben, ansonsten soll eine Meldung erscheinen, dass die Zahl nicht zwischen 1 und 5 liegt. Die Aufgabe ist mit Hilfe der `switch-case`-Anweisung zu realisieren. Das `if`-Konstrukt soll nicht verwendet werden.

Hinweis: Es können auch mehrere ’case’ ’hintereinander’ stehen, also auf die gleiche(n) Anweisung(en) verweisen.

Beispiel 7

(2 Punkte)

Verändere das Beispiel 2 zur Umrechnung von Grad Celsius in Fahrenheit und umgekehrt so, dass ähnlich wie in Aufgabe 5, statt if-Abfragen, ein switch-case-Konstrukt zur Anwendung kommt.

Beispiel 8

(zur Demonstration)

```
//programme: for.c
//author: Josef Strasser-Leitner
//date:  02.10.2005
//version: 1.00
//purpose: demonstration for-loop – calculate square numbers

#include <stdio.h>


// main function
// parameter values: none
// return value: 0
int main()
{
  int loop;
  int quadr;

  for ( loop = 1; loop <= 10; ++loop)
  {
    quadr = loop * loop;
    printf( "Das Quadrat von %d ist %d.\n", loop, quadr);
  }

  return 0;
}
```

Beispiel 9

(7 Punkte)

Schreibe ein Programm, das zwei Zahlen von der Tastatur einliest. Diese Zahlen repräsentieren die Länge und die Breite eines Rechteckes, das dann im Folgenden auf den Bildschirm gezeichnet werden soll. Verwende das * als Zeichen für die einzelnen Punkte (Das Rechteck wird also mit * Zeichen gefüllt).

Tipp: Die Aufgabe kann mit einer inneren und einer äußeren Schleife realisiert werden.

Beispiel 10

(3 Punkte)

Verändere das Beispiel 8 mit der ’for’-Schleife und verwende statt dessen eine ’while’-Schleife. Wir geben auch nur mehr das Quadrat der abgefragten Zahl aus. Mit 0 soll das Programm abbrechen.

```
main()
  {
  int choice;

  choice = 1;
  while (choice != 0)
    {
    printf("Programm zur Berechnung von Quadratzahlen.\n");
    printf("Geben Sie eine Zahl ein ( 0 zum Beenden ): ");

    scanf("%d",&choice);

    // Gib hier die Ausgabe der Quadratzahlen ein!

    } // end while
  } // end main
```

Beispiel 11

(2 Punkte)

Erstelle ein Programm mit einer Schleife, das die ASCII-Zeichen von 32 bis 127 ausgibt.

Verwende den Datentyp `int` für die zu inkrementierende Schleifenvariable und konvertiere den Integer-Wert bei der Ausgabe in `'char'`:

```
int i;
...
printf("%c", (char) i );
...
```

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

Beispiel 12

(6 Punkte)

Die Zahl e wird wie folgt berechnet: e = 1 + 1 / !1 + 1 / ! 2 + 1 / !3 + 1/ !4 + ...
oder anders angeschrieben:

e = 1 + 1 / 1 + 1 / 1 * 2 + 1 / 1*2*3 + 1 / 1*2*3*4 + ...

Wir wollen nun eine Genauigkeitsgrenze festlegen (soll von Tastatur eingelesen werden) und dann das Programm solange einen neuen Nenner nach obigen Muster berechnen lassen, bis die Grenze unterschritten wird:
Der Ablauf des Programms, ist verbal beschrieben, so:

```
Grenze festlegen;
e=1; nenner = 1;

Schleife mit Abbruchbedingung (1 / nenner  <=  grenze);
     {im Schleifenrumpf: e = e + 1 / nenner;
                         Neuen Nenner berechnen;}
```

Anmerkung: In den Schleifen haben wir Laufbedingungen und keine Abbruchbedingungen!

Der neue Nenner berechnet sich am Einfachsten, in dem man den alten Nenner nimmt, und ihn mit der nächsten Zahlenstufe multipliziert, so dass sich diese Folge ergibt:
1.Durchgang: `nenner = nenner * 1;` → 1
2.Durchgang: `nenner = nenner * 2;` → 2
3.Durchgang: `nenner = nenner * 3;` → 6
4.Durchgang: `nenner = nenner * 4;` → 24
Es muss in der Schleife demnach noch eine Zahl `multi` eingeführt werden, die nach jedem Berechnungsschritt um 1 erhöht wird:

Das Ganze noch einmal formuliert:

```
Grenze festlegen;
e=1; nenner = 1;

Schleife mit Abbruchbedingung ( 1 / nenner  <=  grenze);
     {im Schleifenrumpf: e = e + 1 / nenner;
                         multi um 1 erhöhen;
                         nenner = nenner * multi;}
```

Gutes Gelingen!

P.S. Die Genauigkeit 10 hoch minus fünf wird als 1e-5 eingegeben.
(Du kannst natürlich auch 0.00001 schreiben.)