C-ÜBUNGSAUFGABEN FUNKTIONEN-OUTPUT 1

1) Programmabsturz

Warum kann folgendes Programm zu einem Programmabsturz führen ?

```
...
void main(){
  int x;
  int *p;

  x = 5;
  *p = 20;
}
```

Schreiben Sie die folgenden Funktionen und testen diese in einem Programm.

2) Einfache Funktionen

Schreiben Sie die Funktion *verdoppeln*, die eine Zahl verdoppelt:
a) Der Output Parameter wird über return realisiert.
b) Der Output Parameter wird über einen Pointer realisiert. Der Output Parameter und der Input-Parameter ist nicht die gleiche Schnittstellenvariable.
c) Der Output Parameter wird über einen Pointer realisiert. Der Output Parameter und der Input-Parameter wird über die gleiche Schnittstellenvariable realisiert.

3) Maximum und Minimum

Schreiben Sie die Funktion *minimax*, die das Maximum und das Minimum zweier Zahlen bestimmt und außerdem zurückliefert, ob die zwei Zahlen gleich groß sind.

4) Mittelwert

Schreiben Sie die Funktion *mittelwert*, die den arithmetischen Mittelwert und den geometrischen Mittelwert (aber nicht beide zusammen) zweier Zahlen bestimmt und außerdem zurückliefert, ob diese gleich groß sind.
geometrischer Mittelwert von a und b = $\sqrt{ab}$,
arithmetischer Mittelwert von a und b = $\frac{a+b}{2}$

5) Zeichen vertauschen

Schreiben Sie eine Funktion, die zwei Zeichen vertauscht.

6) Zahlen sortieren

a) Schreiben Sie die Funktion vergleiche(...), die zwei Zahlen z1 und z2 der Größe nach vergleicht: $z1 \leq z2$ --> -1, $z1 \geq z2$ --> 1, $z1 = z2$ --> 0
b) Schreiben Sie die Funktion vergleicheStreng(...), die zwei Zahlen z1 und z2 der Größe nach vergleicht: $z1 < z2$ --> -1, $z1 > z2$ --> 1, $z1 = z2$ --> 0

7) Mehrfache einer Zahl

Schreiben Sie eine Funktion, die das Doppelte, das Dreifache und das Vierfache einer Zahl bestimmt.

8) Taschenrechner

Schreiben Sie die Funktion taschenrechner(...), die die Summe, die Differenz, den Quotienten und das Produkt von zwei Zahlen bestimmt.

9) Wo ist der Fehler ?

Die folgende Funktion erzeugt zwar keine Fehlermeldung beim Kompilieren.
Trotzdem ist die Beschreibung und die "Logik" falsch.
Begründen Sie!
Ändern Sie die Funktion so ab, dass alles korrekt wird!

```
/*************************************************************/
/**                                                         **/
/**  int maximum(double z1, double z2, double max)          **/
/**                                                         **/
/*#************************************************************/
/*

Parameter:
  (i) double z1   :  eine Zahl
  (i) double z2   :  eine zweite Zahl
  (o) double max  :  Maximum von z1 und z2

Return:
  (o) Maximum von z1 und z2

Beschreibung:
  Das Programm bestimmt das Maximum von z1 und z2.
*/

int maximum(double z1, double z2, double max){
  if (z1 < z2){
    max = z2;
  }
  else{
    max = z1;
  }
  return(max);
}
```

10)

Schreiben Sie eine Funktion, die das n-fache einer Zahl berechnet.

11) Ersatzwiderstand

Schreiben Sie eine Funktion, die von 3 Widerständen den seriellen und parallelen Ersatzwiderstand berechnet.

12) Potenzrechnung

Schreiben Sie die Funktion potenzen, die das Quadrat und das Kubik zweier Zahlen bestimmt und außerdem zurückliefert, ob das Quadrat und das Kubik gleich groß sind.

13) Bogenmass berechnen

chreiben Sie eine Funktion, die aus dem Gradmass das Bogenmass berechnet.

14) Gradmass berechnen

Schreiben Sie eine Funktion, die aus dem Bogenmass das Gradmass berechnet.

15) Anzahl einer Zahl

Schreiben Sie eine Funktion, die bestimmt, wie oft die erste Zahl a in einer Folge von 5 (allgemein n) ganzen Zahlen vorkommt.

16) Zahlen eines Feldes sortieren

Schreiben Sie eine Funktion, die ein aus Zahlen bestehendes Feld der Länge 2 der Größe nach sortiert.

17) Zahlen eines Feldes vervielfachen

a) Schreiben Sie eine Funktion, die alle Elemente eines Feldes verdoppelt.
b) Schreiben Sie eine Funktion, die alle Elemente eines Feldes n-fach vervielfacht.

18) sehr schwer und aufwendig

Die Addition, Subtraktion, Multiplikation und Division zweier ganzen Zahlen des Datentyps int bzw. long ist leider nur innerhalb eines bestimmten Datenbereichs möglich.
Erstellen Sie eine Funktion, die diese elementaren Operationen innerhalb eines beliebig groen Datenbereichs macht (Die Ziffern der Zahlen werden als Zeichen in einem Feld dargestellt)

19) schwer und aufwendig

Erstellen Sie eine Menge von Funktionen, die Geraden im zweidimensionalen Raum betreffen:
Schnittpunkt(e) zweier Gerade, Senkrechte zu einer Geraden bestimmen, Zweipunkteform einer Geaden, Punkt-Steigungsform einer Geraden, usw.

20) Maximum und Minimum bestimmen

Erstellen Sie eine Funktion, die das Maximum und Minimum der Zahlen eines Feldes bestimmen.

21) Kapital anlegen

Ein Anfangskapital wird zu einem bestimmten Zinssatz (auf einen Zeitabschnitt wie z.B. ein Jahr).bezogen eine bestimmte Anzahl Zeitabschnitte (z.B. Jahre) auf einem Konto angelegt, ohne die Zinsen abzuheben.
Schreiben Sie eine Funktion, die das Endkapital, die insgesamt angelaufenen Zinsen und den Faktor bestimmt, um den sich das Anfangskapital vervielfacht hat.

22)

Schreiben Sie eine Funktion, die die Strafen ermittelt, die sich bei einer Geschwindigkeitsüberschreitung ergeben (Strafkatalog siehe Internet).
Für Internetbenutzung vorher um Erlaubnis fragen.

23)

Die folgende Funktion p(n) ermittelt die n-te Primzahl

$$p(n) = \sum_{m=1}^{2^n} \left[\sqrt[n]{n} \cdot \left(\sum_{k=1}^{m} \left[(\cos^2(\pi \frac{(k-1)!+1}{k})) \right] \right)^{-\frac{1}{n}} \right]$$

Bem: Die eckige Klammer bedeutet die Abrundungsfunktion
Beispiele: p(1) = 2, p(2) = 3, p(3) = 5
[2,8] = 2 [3,1415] = 3

24) Näherungsweise Flächenberechnung (Normalparabel)

Berechnen Sie die angenäherte Fläche zwischen den Geraden x = a, x = b, der x-Achse und der Kurve mit dem Schaubild $f(x) = x^2$, indem Sie diese Fläche durch die Summe von Rechteckflächen annähern:

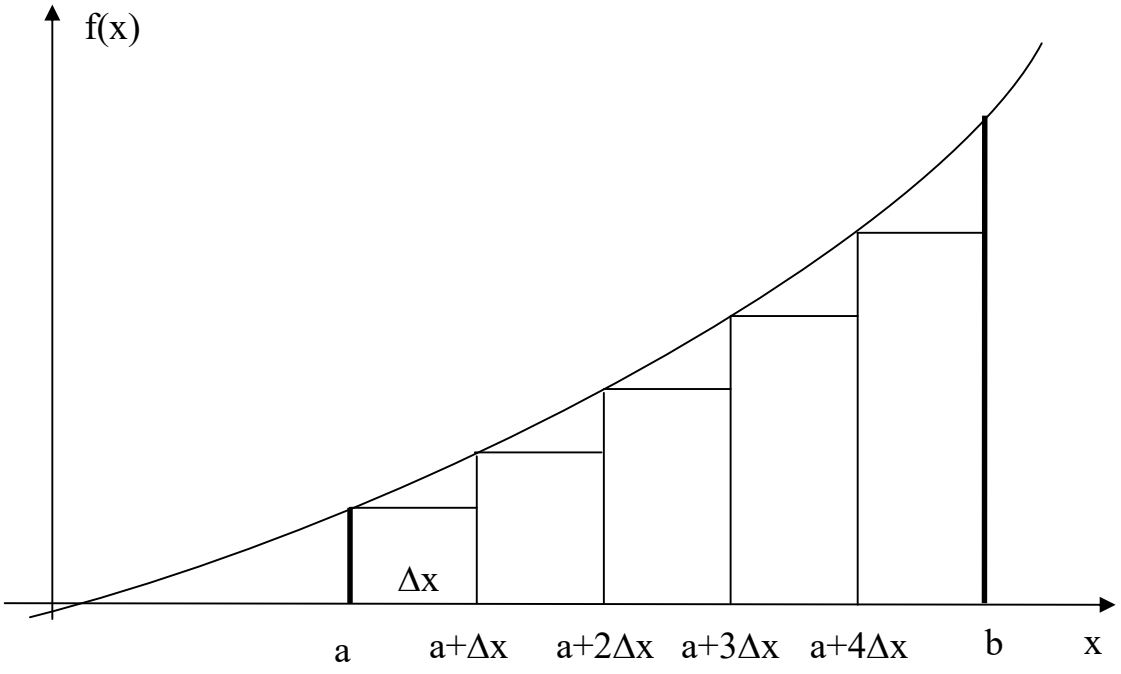
$$\Lambda x = \frac{b-a}{n}$$

Wenn man das Intervall [a ; b] z.B. in n = 5 gleich lange Intervalle mit der Breite $\Lambda x = \frac{b-a}{n}$ unterteilt, dann ist die Summe der Flächen der Rechtecke:

$A = f(a)\cdot\Delta x + f(a+\Delta x)\cdot\Delta x + f(a+2\Delta x)\cdot\Delta x + f(a+3\Delta x)\cdot\Delta x + f(a+4\cdot\Delta x)\cdot\Delta x$

allgemein gilt (für beliebiges n):

$A = f(a)\cdot\Delta x + f(a+\Delta x)\cdot\Delta x + f(a+2\Delta x)\cdot\Delta x + f(a+3\Delta x)\cdot\Delta x + \ldots + f(a+(n-1)\cdot\Delta x)\cdot\Delta x$



Zusatzaufgabe:
Nähern Sie die Fläche nicht durch Rechtecke, sondern durch Trapeze an.

25) Funktionswert berechnen

Schreiben Sie die Funktion p, die von einer Polynominalfunktion

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \ldots + a_n \cdot x^n$$

den Wert an einer beliebigen Stelle x berechnet.

26) Exakte Flächenberechnung

Schreiben Sie eine Funktion, die den exakten Wert der bilanzierten Fläche berechnet, die von einem Polynom, der x-Achse und zwei zu der y-Achse parallelen Gerade begrenzt wird.

27) Näherungsweise Flächenberechnung

Schreiben Sie eine Funktion, die eine Näherung der bilanzierten Fläche berechnet, die von einem Polynom, der x-Achse und zwei zu der y-Achse parallelen Gerade begrenzt wird.
Die Näherung wird durch Treppen (Rechtecke unter der Kurve) realisiert.

28) Collatz-Vermutung

Es geht um Zahlenfolgen, die nach dem folgenden Bildungsgesetz formuliert werden:
Beginne mit einer ganzen Zahl n > 0
Ist die Zahl gerade, dann dividiere sie durch 2
Ist die Zahl ungerade, dann multipliziere sie mit 3 und addiere 1 dazu.
Wiederhole die Vorgehensweise mit der erhaltenen Zahl

Collatz-Vermutung (siehe Spektrum der Wissenschaft Februar 2014)
Jede so konstruierte Collatz-Zahlenfolge mündet in den Zyklus 4, 2, 1, egal, mit welcher natürlichen Zahl n > 0 man beginnt.

Bem:
Die Collatz-Vermutung ist bis heute unbewiesen.

Beispiel:
5 --> 3 · 5 + 1=16 --> 16 / 2 = 8 --> 8 / 2 = 4 --> 4 / 2 = 2 --> 2 / 2 = 1 (Ende)

Schreiben Sie eine Funktion, die eine ganze Zahl n > 0 die Collatz-Zahlenfolge in ein Feld (oder eine Datei) schreibt.

29) Conways Formel (Powertrain)

n sei die positive Dezimalzahl mit der Ziffernfolge abcdef ....
Daraus wird die folgende Dezimalzahl konstruiert: $a^b c^d e^f ...$
Besteht n aus einer ungeraden Anzahl von Ziffern, wählt man die 1 als Hochzahl der letzten Ziffer.
Beispiel:
3462 ==> $3^4 6^2 = 81 * 36 = 2916$. Diese Zahl wird wieder "conwayd":
2916 ==> $2^9 1^6 = 512 * 1 = 512$ ==> $5^1 2^1 = 5*2 = 10$ ==> $1^0 = 1$

Aufgabe:
Suchen Sie eine maximal 4-stellige Zahl, deren Conway genau wieder die gleiche Zahl liefert. (Diese Zahl nennt man einen Fixpunkt).

30)

a) Schreiben Sie eine Funktion, die das am häufigsten vorkommende Zeichen in einer Zeichenfolge bestimmt.
Wenn es mehrere Zeichen gibt, die am gleich häüfigsten vorkommen, soll eine davon bestimmt werden

b) Wenn es mehrere Zeichen gibt, die am gleich häüfigsten vorkommen, sollen diese ebenfalls bestimmt werden

31) Welche Strategie ist erfolgreicher ?

Es gibt zwei Strategien, um in einem Spiel einen Gewinn zu machen:
Strategie_2:
Sobald das erste Mal "Kopf" beim Werfen einer Münze erscheint, hat man gewonnen.
Maximal zweimal darf die Münze geworfen werden.
Strategie_6:
Sobald das erste Mal die "Sechs" beim Werfen einesWürfels erscheint, hat man gewonnen.
Maximal sechsmal darf gewürfelt werden.
In welchem Spiel ist die Gewinnwahrscheinlichkeit höher ?
Schreiben Sie die 2 Funktionen strategie_2() und strategie_6()
Diese liefern zurück, ob die Strategie erfogreich war oder nicht (Münzwürfe und Werfen eines Würfels werden durch Zufallszahlen simuliert).
Schreiben Sie dann die Funktion testen(int anzahl), in der diese 2 Funktionen hinreichend oft (z.B. 1000000 Mal) aufgerufen werden und festgestellt wird, wie oft jede erfolgreich.
Die Funktion testen(int anzahl) gibt dann zurück, welche Strategie erfolgreicher ist, bzw. ob sie gleich erfolgreich sind.

32) Ein Feld mit Zufallszahlen erzeugen (Permutation)

gegeben ist eine ganze Zahl n > 0
Erzeugen Sie eine zufällige Reihenfolge der Zahlen 0 bis n-1 und speichern dise in einem Feld.
Beispiel:
n = 5
zufällige Zahlenfolge = 3, 0, 4, 1, 2

Tipp:
Belege ein Feld v mit den Zahlen 0, 1, 2, ... n-1
und erzeuge eine "leeres" Feld w.
v: 0, 1, 2, 3, 4
Schritt1:
Berechne zufälligen Index i = rand % 5, ergibt z.B. = 0
Füge v[0] ans Feldende von w, also:
w: 0
Lösche v[0], also v = 1, 2, 3, 4
Schritt2:
Berechne zufälligen Index i = rand % 4, ergibt z.B. = 2
Füge v[2] ans Feldende von w, also:
w: 0, 3
Lösche v[2], also v = 1, 2, 4
usw.

33) Tilgungsplan

Entwickeln Sie eine Funktion für einen Tilgungsplan:
Nach Angabe von Kreditsumme, Darlehnszinsen pro Jahr (umrechnen in Darlehnszinsen pro Monat, siehe Rechenbeispiel unten) und monatlicher Rate sollen die Dauer der Tilgung und die Gesamtkosten (Summe aller Zinsen, die an die Bank gezahlt wurden !) berechnet werden. Die monatliche Rate wird zur Zinszahlung und Tilgung benutzt.

Mathematische Anleitung anhand eines Beispiels:
Voraussetzungen: Kreditsumme S = 1000 EURO, Darlehnszinsen pro Jahr p = 120%, d.h. 120/12 = 10% Darlehnszinsen pro Monat, monatliche Rate = 500 EURO.

n	Zinsen z_n	Tilgung tg_n	Schulden S_n
0	0	0	1000
1	1000 · 0,1 = 100	500 - 100 = 400	1000 - 400 = 600
2	600 · 0,1 = 60	500 - 60 = 440	600 - 440 = 160
3	160 · 0,1 = 16	160	0

Im 3. Monat tilgt der Kunde 160 Euro und zahlt noch 16 Euro Zinsen. Deshalb bekommt er noch 500 - (16 + 160) = 324 Euro zurück.
Insgesamt hat man an die Bank 100 + 60 + 16 = 176 Euro Zinsen gezahlt.

34) schwer

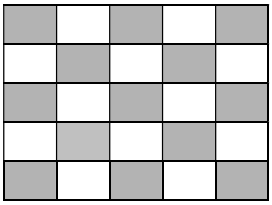
Schreiben Sie eine Funktion, die die Lösung(en) der quadratischen Gleichung:
$a\,x^2 + b\,x + c = 0$ (a, b, c sind beliebige reelle Zahlen)
ermittelt.

35) schwer

Schreiben Sie eine Funktion, die die Lösung eines LGS mit 3 Gleichungen und 3 Unbekannten liefert.

36) Springer im Schachfeld (mit brute force oder nachdenken)

In einem 5 x 5 Schachbrett ist jedes Feld mit genau einem Springer besetzt. Mit allen 25 Springern soll gleichzeitig ein Springer-Zug gemacht werden. Danach muß wieder auf jedem Feld des Schachbretts genau ein Springer stehen.
Ist das möglich ? Wie muß jeder einzelne Springer ziehen ?



37)

Schreiben Sie eine Funktion, die eine ganze Zahl in ihre Primfaktoren zerlegt.
Beispiel:
$450 = 2 * 3^2 * 5^2$

38)

Eine Pyramide soll je nach Benutzereingabe erstellt werden.
Diese Benutzereingabe soll die Basis für die Pyramide sein:
Beispiel:
Benutzereingabe "H":

```
        A
       ABA
      ABCBA
     ABCDCBA
    ABCDEDCBA
   ABCDEFEDCBA
  ABCDEFGFEDCBA
ABCDEFGHGFEDCBA
```

39) Pascalsches Dreieck

Um die Formel $(a+b)^n$ umzuformen, kann dies mit Hilfe des Pascalschen Dreiecks gemacht werden.
Beispiel:
$(a+b)^4 = \mathbf{1}\cdot a^4 + \mathbf{4}\cdot a^3b^1 + \mathbf{6}\cdot a^2b^2 + \mathbf{4}\cdot a^1b^3 + \mathbf{1}\cdot b^4$
Die Koeffizienten 1, 4, 6, 4, 1, erhählt man aus dem Pascalschen Dreieick, das wie folgt aufgebaut ist:

```
              1
            1   1
          1   2   1
        1   3   3   1
      1   4   6   4   1
    1   5   10  10  5   1
  1   6   15  20  15  6   1
.................................
```

Die erste und letzte Zahl einer jeden Zeile im Pascalschen Dreieck ist immer die 1.
Die anderen Zahlen einer Nachfolger-Zeile berechnen sich aus der Vorgängerzeile, in dem dort jeweils die Summe der benachbarten Zahlen berechnet werden.
Beispiel:

```
1   4   6   4   1
```

Daraus berechnet man
1 + 4 = 5
4 + 6 =10
6 + 4 = 10
4 + 1 = 5

Schreiben Sie eine Funktion, die n-te Zeile eines Pascalschen Dreiecks berechnet.

40) Laufschrift V3

Schreiben Sie ein Programm (mit Hilfe von Funktionen), das eine Laufschrift auf dem Bildschirm erzeugt: Es soll der über mehrere Zeilen verteilte Großbuchstabe V von links nach rechts (und dann nach unten) "über" den den Bildschirm "laufen".

41) Tic-Tac-Toe

Implementieren Sie ein Programm, mit dem man gegen den Computer Tic-Tac-Toe spielen kann.

42) Spielpaarungen und Spieltage erstellen (z.B. für die Fußballbundesliga)

In der 1. Fußballbundesliga gibt es insgesamt 18 Mannschaften. Jede dieser Mannschaften muß in 17 Spieltagen gegen jede andere Mannschaft genau einmal gespielt haben, ohne dabei an einem Spieltag auszusetzen.
Implementieren Sie ein Programm, das von n (gerade Anzahl) Mannschaften für die n-1 Spieltage die Mannschaftspaarungen ausgibt bzw. in ein Feld schreibt.

Bemerkungen:
1) Beispiel für n=4
1. Spieltag: (1,2), (3,4)
2. Spieltag: (1,3), (2,4)
3. Spieltag: (1,4), (2,3)

2) Für z.B. n = 6 kann man in einer mehrfach verschachtelten Schleife alle möglichen 6-er Paare (= eventuelle Spieltage) erzeugen:
(1, 1, 1, 1, 1, 1) kein Spieltag
(1, 1, 1, 1, 1, 2) kein Spieltag
…
(1, 2, 3, 4, 5, 6) ein Spieltag
…
(6, 6, 6, 6, 6, 6) kein Spieltag

wobei z.B. (1, 2, 3, 4, 5, 6) bedeutet, daß folgende Mannschaftspaarungen an diesem Spieltag stattfinden: (1,2) , (3,4) , (5,6)
und (1, 1, 1, 1, 1, 2) bedeutet z.B. ein nicht korrekter Spieltag, der dann programmtechnisch ignoriert bzw. herausgefiltert wird.
In dieser mehrfach verschachtelten Schleife wid dann in der innersten Schleife geprüft, ob der erzeugte Spieltag korrekt ist und ob er noch nicht in dem bis dahin erzeugten Feld vorkommt. Dies ist allerdings ein Algorithmus mit schlechtem Laufzeitverhalten.

3) schwer
Entwickeln Sie einen Algorithmus mit einem besseren Laufzeitverhalten.

43) Epidemie (Corona) simulieren

Zur Information:
Es soll ein mathematisches Modell einer Epidemie einer kg-Population (d.h. einer Population, die am Anfang aus k kranken und g gesunden Personen besteht) entwickelt und dann durch ein Programm simuliert werden. Es können gesunde auf gesunde, kranke auf kranke und kranke auf gesunde Personen treffen. Mit einer bestimmten Wahrscheinlichkeit (WK) werden dann diese Personen krank bzw. gesund.
k: Anzahl kranker Personen
g: Anzahl gesunder Personen
n = k+g = Anzahl der Personen der Population

Es gibt also 3 Fälle:

Fall1: Beide Personen sind gesund (GG)
Es passiert nichts.

Fall2: Beide Personen sind krank (KK)
Für jede der beiden Personen gilt:
Mit WK = 1/6 stirbt die Person.
Mit WK = 1/6 wird die Person wieder gesund.
Mit WK = 2/3 bleibt die Person krank.

Fall3: Eine Person ist krank und die andere ist gesund (KG)
Für die gesunde Person gilt:
Mit WK = 1/2 wird die Person krank.
Für die kranke Person gilt:
Mit WK = 1/6 stirbt die Person.
Mit WK = 1/6 wird die Person wieder gesund.
Mit WK = 2/3 bleibt die Person krank.

a)
Nach jedem Simulationsschritt muss die Anzahl der gesunden und kranken Spieler (und die Summe davon) auf dem Bildschirm ausgegeben werden.

44) Acht-Damen-Problem

Zur Information:
Auf einem 8x8-Schachbrett sollen 8 Damen so verteilt werden, daß sie sich nicht bedrohen.
Wie viele Stellungen (ohne gegenseitige Bedrohungen) gibt es?

Die rechten Diagonalen 0 bis 14 des 8x8-Schachbretts

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8
2	3	4	5	6	7	8	9
3	4	5	6	7	8	9	10
4	5	6	7	8	9	10	11
5	6	7	8	9	10	11	12
6	7	8	9	10	11	12	13
7	8	9	10	11	12	13	14

Die linken Diagonalen 0 bis 14 des 8x8-Schachbretts

7	8	9	10	11	12	13	14
6	7	8	9	10	11	12	13
5	6	7	8	9	10	11	12
4	5	6	7	8	9	10	11
3	4	5	6	7	8	9	10
2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

Aufgaben
Tipp: Dame gesetzt: 1 keine Dame: 0

a) Schreiben Sie die Funktion
`initFeld(int[][] feld)`
die alle Elemente eines 8x8-Schachbretts auf 0 (also keine Dame) setzt.

b) Schreiben Sie die Funktion
`printFeld(int[][] feld) {`
die alle Elemente eines 8x8-Schachbretts in 8x8-Format auf Bildschirmausgibt.

c) Schreiben Sie die Funktion
`int summeRD(int nr, int feld[][])`
die die Anzahl aller Damen auf einer rechten Diagonale eines 8x8-Schachbretts berechnet.

d) Schreiben Sie die Funktion
`int summeRD(int nr, int feld[][])`
die die Anzahl aller Damen auf einer rechten Diagonale eines 8x8-Schachbretts berechnet.

e) Schreiben Sie die Funktion
`int summeLD(int nr, int feld[][])`
die die Anzahl aller Damen auf einer linken Diagonale eines 8x8-Schachbretts berechnet.

f) Schreiben Sie die Funktion
`int summeZeile (int nr, int feld[][])`
die die Anzahl aller Damen einer Zeile eines 8x8-Schachbretts berechnet.

g) Schreiben Sie die Funktion
`int summeSpalte (int nr, int feld[][])`
die die Anzahl aller Damen einer Spalte eines 8x8-Schachbretts berechnet.

e) Schreiben Sie die Funktion
`bool istSchachbrettOhneBedrohung (int[][] feld)`
die berechnet, ob es auf einem 8x8-Schachbrett keine Damen gibt, die sich bedrohen

f) Erzeugen Sie in main() alle möglichen Stellungen von 8 Damen auf einem 8x8-Schachbrett. Geben Sie die bedrohungsfreien Stellungen (in denen es keine gegenseitige Bedrohungen durch Damen gibt) auf dem Bildschirm aus,.
Geben Sie die Anzahl der bedrohungsfreien Stellungen auf dem Bilschirm aus.

C-ÜBUNGSAUFGABEN FUNKTIONEN-OUTPUT 2

45) Spiel "Black Jacket"

1) Ihr Bekannter Herr X hat im Internet von einer Spielstrategie erfahren, mit der er mit dem Spiel "Black Jacket" im Spielcasino erfolgreich sein soll. Als vorsichtiger Mensch raten Sie ihm, dieses Vorhaben zuerst mal am Computer zu simulieren, bevor er im Spielcasino viel Geld verliert.

Zur Information:
Das Spiel "Black Jacket" ("17 + 4") funktioniert wie folgt beschrieben:
Das Ziel eines jeden Spielers ist es, möglichst nahe an die 21, aber nicht über die 21 zu kommen.
Spieler_1 "würfelt" so lange mit einem Spezialwürfel (mit dem genau eine der folgenden Zahlen 2, 3, 4, 7, 8, 9, 10, 11 erwürfelt werden kann), bie er aufhören will.
Die einzelnen Würfe sind verdeckt und vom Gegenspieler nicht zu sehen.
Dann macht der Gegenspieler Spieler_2 die gleiche Prozedur.
Wer die größere Summe seiner Würfe hat, hat gewonnen.
Wer eine Summe > 21 hat, hat verloren (wenn die Summe des Gegeners <22).
Bei gleicher Summe ist der Spielausgang unentschieden.
Hat jeder Spieler eine Summe > 21, ist der Spielausgang ebenfalls unentschieden.
Wenn alle 2 Spieler nicht mehr würfeln wollen (bzw. maximal 10 mal gewürfelt haben), wird das Spiel sofort beendet und auf dem Bildschirm die entsprechenden Infos ausgegeben (Gewinner und Verlierer mit jeweiliger Punktzahl).

Weisen Sie jedem Spieler eine Spielstrategie zu (z.B. hört nach 2 Würfen auf).
Simulieren Sie eine bestimmte Anzahl Spiele (Spieler_1 gegen Spieler_2) und berechnen Sie den Prozentsatz gewonnener Spiele von Spieler_1.
Damit kann man entscheiden, welche Strategie erfolgreicher ist.

2) Black Jacket wird eigentlich mit Skatkarten gespielt.
Die Skatkarten bestehen aus 32 Einzelkarten und sind dabei in vier Farben zu je acht Karten unterteilt. Jede Farbe enthält acht Karten, die jeweils einen Wert haben.
Sieben (7), Acht (8), Neun (9), Zehn (10), Bube (2), Dame (3), König (4) und Ass (11).
Die 32 Skatkarten liegen in Form eines Stapels auf dem Tisch.
Für ein Spiel wie z.B. Black Jack ("17 + 4") werden Karten von diesem Stapel genommen.
Simulieren Sie diesen Kartenstapel durch die Klasse "Kartenstapel" und spielen damit Black Jacket.

46) Zeichen-Felder verwalten

Implementieren Sie die wie folgt dokumentierten Funktionen:

```
/*****************************************************************/
/**                                                             **/
/**  int ueberschreibe(int pos, int anz, char z, char string[])  **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i)   int pos: Stelle im Feld string
  (i)   int anz: Anzahl der zu überschreibenden Zellen im Feld
                 string
  (i)   char z : Zeichen
  (i/o) char string[] != leer : Zeichenfolge

Return:
  0 : alles ok
  -1:  Falls pos außerhalb des Feldes liegt oder
       Stellen überschrieben werden würden, die sich außerhalb
       des Feldes befinden.

Beschreibung:
  Überschreibt ab der Stelle pos (einschließlich) im Feld string
  insgesamt anz Zeichen mit dem Zeichen z.
  Bem: Dem ASCII_Zeichnen mit dem Wert 255 folgt das ASCII-
       Zeichen mit dem Wert 0

*/


/*****************************************************************/
/**                                                             **/
/**  void kodieren(char string[])                               **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i/o) char string[] != leer : Zeichenfolge

Return:
  kein

Beschreibung:
  Kodiert das Feld string indem jedes Zeichen in das in der
  ASCII-Tabelle folgende umgewandelt wird.

Beispiel:
  "aber"   --->   kodieren   --->   "bcfs"
*/
```

```
/*****************************************************************/
/**                                                             **/
/**  void dekodieren(char string[])                             **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i/o) char string[] != leer : Zeichenfolge

Return:
  kein

Beschreibung:
  dekodiert das Feld string indem jedes Zeichen in das
  in der in der ASCII-Tabelle vorgehende umgewandelt wird.
  Bem: Dem ASCII_Zeichnen mit dem Wert 255 folgt das ASCII-
       Zeichen mit dem Wert 0

Beispiel:
  "bcfs"   --->   dekodieren   --->   "aber"
*/


/*****************************************************************/
/**                                                             **/
/**  void sortieren(int modus, char string[])                   **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i)    int modus: (0: aufsteigend, 1: absteigend)
  (i/o) char string[] != leer : Zeichenfolge

Return:
  kein

Beschreibung:
  Sortiert ein Feld aufsteiegend (modus=0) oder absteiegend
  (modus=1).
*/

/*****************************************************************/
/**                                                             **/
/**  bool istVorhanden(char z, char string[])                   **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i) char z : Zeichen
  (i) char string[] != leer : Zeichenfolge

Return:
  true : z kommt in string vor
  false: z kommt nicht in string vor

Beschreibung:
  Prüft nach, ob z in einer Zeichenfolge vorkommt.
*/
```

```
/************************************************************/
/**                                                        **/
/**  int suche(char string[], char splitter[])             **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) char string[] != leer   : Zeichenfolge
  (i) char splitter[] != leer : Zeichen, das die Zeichenfolge
                                splittet

Return:
  Gibt den Index des gefundenen Splitters zurück (falls einer
  gefunden wird). Sonst wird -1 zurückgeliefert.

Beschreibung:
  Es wird der Trenner aus der Menge splitter[] gesucht, der
  das 1. Mal in der Zeichenfolge "string" auftaucht.
  Dessen Index wird zurückgegeben.
  Falls kein Trenner aus der Menge splitter[] in der Zeichenfolge
  "string" auftaucht, wird der Index -1 zurückgegeben.
*/


/************************************************************/
/**                                                        **/
/**  int deleteChar(char string[], int index)              **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i/o) char string[] != leer : Zeichenfolge
  (i)   int index >= 0 :      : An dieser Stelle wird das Zeichen
                                entfernt.

Return:
  Gibt -1 zurück, falls der Index außerhalb der feldgrenzen liegt.
  sonst 0

Beschreibung:
  In der Zeichenfolge string wird an der Stelle index das zugehörige
  Zeichen entfernt.

Beispiel:
  char string[20] = "abcdefg";
  index = 0;

  r = deleteChar(string, index){
  liefert:
  string = "bcdefg"
  r = 0
*/
```

```
/*****************************************************************/
/**                                                             **/
/**  int splits2(char string[], char splitter[], char splitter  **/
/**                              char str1[], char str2[])      **/
/**                                                             **/
/*#****************************************************************/
/*
Parameter:
  (i) char string[] != leer   : Zeichenfolge
  (i) char splitter[] != leer : Zeichen, das die Zeichenfolge
                                splittet
  (o) char str1[]             : Zeichenfolge in string links des
                                Splitters
  (o) char str2[]             : Zeichenfolge in string rechts des
                                Splitters

Return:
  Gibt den Index des gefunden Splitters zurück (falls einer
  gefunden wird). Sonst wird -1 zurückgeleifert.

Beschreibung:
  Der Trenner aus der Menge splitter[], der das 1. Mal in der
  Zeichenfolge "string" auftaucht, zerlegt die Zeichenkette
  "string" in die Teilzeichenfolge str1[] links des Trenners und
  str2[] rechts des Trenners.
  Falls kein Trenner aus der Menge splitter[] in der Zeichenfolge
  "string" auftaucht, wird str1 und str2 gleich der leeren
  Zeichenfolge gesetzt.

Beispiel:
  char string[20] = "((a+b)*((c+d)*x))";
  char splitter[5] = "/+-*";
  index = splits2(string, splitter, str1, str2);

  liefert:
  index = 3
  str1 = "((a"
  str2 = " b)*((c+d)*x))"
*/
```

```
/************************************************************/
/**                                                        **/
/**  void sortieren(int feld[], int i1, int i2, int modus)   **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i/o) int feld[] :  Feld, das aus Zahlen besteht.
  (i) int i1       :  kleinere Grenze
  (i) int i2       :  größere Grenze
  (i) int modus: 0 (aufsteigend) und 1(absteigend)

Return:
  kein

Beschreibung:
  Sortiert eine Folge von Zahlen zwischen den Indizes (Stellen)
  i1 und i2 nach dem Modus aufsteigend oder absteigend.

Beispiel:
  int feld [6] : 6 5 6 3 7 1
  i1 = 1
  i2 = 4
  modus = 0

  liefert:
  feld [6] : 6 3 5 6 7 1
*/


/************************************************************/
/**                                                        **/
/**  void anfuegen(int feld[], int zahl)                   **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i/o) int feld[]   :  Feld, das aus Zahlen besteht.
  (i) int zahl       :  Zahl, die an das Feldende angefügt wird.

Return:
  kein

Beschreibung:
  Wenn die Zahl "zahl", nicht in dem Array "feld" vorkommt, wird
  diese an das Feldende angefügt.
  Wenn die Zahl "zahl", in dem Array "feld" vorkommt, geschieht
  nichts.
  Trick: Im Element feld[0] wird die Länge des Feldes gespeichert.
  Diese Länge wird automatisch um eins erhöht, wenn eine Zahl
  angefügt wird. In feld[0] darf als keine "nornmale" Zahl
  gespeichert werden, sondern wird intern die Feldlänge verwaltet.

Beispiel:
  int feld [6] : 2 9 12 6 7 9
  anfuegen(feld, 1)

  liefert:
  feld [6] : 3 9 12 1 7 9
*/
```

47) Zeichen-Felder verwalten (Fortsetzung)

Schreiben Sie Funktionen zur Stringmanipulation:
Anfügen, Ersetzen, usw. eines Strings (siehe unten).
Benutzen Sie als Vorbild die Funktionen der Entwicklungsumgebung, wie z.B.
strcmp, strcpy, strcat, usw.

Beispiele:

`strcmp(...)`
vergleicht (engl: string compare) 2 Zeichenfolgen lexikografisch (wie z.B. zwei Namen im Telefonbuch).

`concat(...)`
Eine Zeichenfolge wird an die andere angefügt (engl: concatenate).

`contains(...)`
prüft nach, ob eine Zeichenfolge eine andere beinhaltet (engl: contain).

`endsWith(...)`
prüft nach, ob eine Zeichenfolge mit einer bestimmten Zeichenfolge endet.

`indexOf(...)`
Berechnet den Index (das erstmalige Vorkommen) eines Zeichens in einer Zeichenfolge.

`lastIndexOf(...)`
Berechnet den Index (das letztmalige Vorkommen) eines Zeichens in einer Zeichenfolge.

`length(...)`
berechnet die Länge einer Zeichenfolge.

`deleteFirstChar(...)`
Löscht in einer Zeichenfolge das erste Auftreten eines bestimmten Zeichens

`deleteAllChars(...)`
Löscht in einer Zeichenfolge jedes Auftreten eines bestimmten Zeichens

`deleteFirstString (...)`
Löscht in einer Zeichenfolge das erste Auftreten einer bestimmten Zeichenfolge

`deleteAllString (...)`
Löscht in einer Zeichenfolge jedes Auftreten einer bestimmten Zeichenfolge

`replaceFirstChar(...)`
Ersetzt in einer Zeichenfolge das erste Auftreten eines bestimmten Zeichens durch eine bestimmte Zeichenfolge.

`replaceAllChars(...)`
Ersetzt in einer Zeichenfolge jedes Auftreten eines bestimmten Zeichens durch eine bestimmte Zeichenfolge.

`replaceFirstString(...)`
Ersetzt in einer Zeichenfolge das erste Auftreten einer bestimmten Zeichenfolge durch eine bestimmte Zeichenfolge.

`replaceAllStrings (...)`
Ersetzt in einer Zeichenfolge jedes Auftreten einer bestimmten Zeichenfolge durch eine bestimmte Zeichenfolge.

`replace(...)`
Ersetzt in einer Zeichenfolge jedes Auftreten eines bestimmten Zeichens durch ein anderes Zeichen.

`split2(...)`
teilt eine Zeichenfolge in 2 Teilzeichenfolgen, wobei das erstmalige Vorkommen eines bestimmten Zeichens die erste Teilzeichenfolge angibt.
Beispiel:
Zeichenfolge: "abcasdfgfdgdf"
Begrenzer: ' f '
Die zwei Teilzeichenfolgen:
"abcasd" und "gfdgdf"

`splits2(char str[], char begrenzer[], int i)`
ähnlich wie split(..), nur komplexer.
Beispiele:
str: ((a+b)*c)+(d*b))
begrenzer: +*
splits2(str, begrenzer, 1) liefert: " ((a" und "b)*c)+(d*b))"
splits2(str, begrenzer, 2) liefert: "((a+b)" und " c)+(d*b))"
splits2(str, begrenzer, 4) liefert: "((a+b)*c)+(d" und "b))"

`startsWith(...)`
prüft nach, ob eine Zeichenfolge mit einer bestimmten Zeichenfolge beginnt.

`substring(...)`
berechnet die Teilzeichenfolge einer Zeichenfolge, die an einem bestimmten Index beginnt.
Beispiel:
"abcdefghabc", 5 -->"fghabc"

`substringBetween(...)`
berechnet die Teilzeichenfolge einer Zeichenfolge, die an einem bestimmten Index beginnt und an einem anderen Index endet.
"abcdefghabc", 5, 7 --> "gha"

Bemerkung:
Alle die vorigen Funktionen, die mit Feldern implementiert wurden, kann man auch mit während der Laufzeit reserviertem Speicher (mit malloc) realisieren und hat damit wieder neue Aufgaben.

48) Zahlen-Felder verwalten

Wenn ein Feld erstellt wird, hat es eine bestimmte vorgegebene Länge maxLen.
Da in einem Zahlenfeld das Ende des Feldes nicht wie in einem Zeichenfeld mit '\0' gekennzeichnet ist, wird das letzte Elemente der Feldes (letzte zahl) hier anders vermerkt:
An der Stelle 0 des Feldes wird immer die Stelle des letzten Elements des Feldes gespeichert
Wenn das Feld erzeugt wird, hat dieser Zeiger den Wert -1
Nach jedem An - oder Einfügen wird der Wert dieses Zeigers um 1 ehöht, bei jedem Löschen eines Elements aus dem Feld wird der Zeigerwert um 1 verringert.
Der Wert dieses Zeigers (auf das letzte Element) darf maximal maxLen-1 sein.
Dies muß durch die einzelnen Funktionen garantiert werden.

Zur Feldverwaltung sollen folgende Methoden gehören:

```
void anfuegen(int v[], int zahl);
```

fügt die Zahl "zahl" an das Ende des Fedes "v" an, vorausgesetzt, daß nicht die Länge (maxLen) des Feldes überschritten wird.

Wichtig:
An der Stelle 0 desFeldes wird immer die aktuelle Länge des Feldes eingetragen und verwaltet.

Beispiel: maxLen = 8

4	12	132	17	29	?	?	?

anfügen der Zahl 13:

5	12	132	17	29	13	?	?

```
void druckeFeld(int v[])
```

gibt alle Elemente des Feldes auf dem Bildschirm aus.

```
void entferneLetztesElement(int v[])
```

entfernt das letzte Element des Feldes, wobei die Länge des Feldes angepaßt (um 1 verringert) wird.

```
void entferneAnPosition(int v[], int pos)
```

Lösche das Element an der Stelle pos
Beispiel:
Feld: 4 5 9 1 3
Lösche Element an der Stelle 2
Feld: 4 5 1 3

`void einfuegenRechts(int v[] , int pos, int zahl)`
Für pos muß gelten: -1<=pos <=zeigerLetztesElement
die Zahl zahl wird rechts von pos, also an der Stelle pos+1 eingefügt (nicht überschrieben)
Vorher werden die entsprechenden Zahlen noch nach rechts verschoben.
Beispiel:
Feld: 7 5 9 3
Einfügen der Zahl 8 an rechts von pos = 2 ergibt:
Feld: 7 5 9 8 3

`void einsortierenRechts(int v[], int zahl)`
Fügt das Element zahl rechts von der Stelle ein, wo zahl das 1. Mal vom Wert her größer (oder gleich) ist als das dort sich befindliche Feldelement.
Ansonsten wird es rechts von der Stelle -1 eingefügt.
Beispiel:
Feld: 4 5 9 1 3
Rechts einsortieren der Zahl 8 ergibt:
Feld: 4 5 8 9 1 3

`void einsortierenLinks(int v[], int zahl)`
Fügt das Element zahl links von der Stelle ein, wo zahl das 1. Mal vom Wert her größer (oder gleich) ist als das dort sich befindliche Feldelement.
Ansonsten wird es an das Feldende angefügt.
Beispiel:
Feld: 4 5 9 1 3
Links einsortieren der Zahl 2 ergibt:
Feld: 4 5 9 2 1 3

`void entferneZahl(int v[], int zahl)`
wenn die Zahl zahl das erste Mal im Feld vorkommt, wird sie entfernt.
Falls sie nicht vorkommt, passiert nichts.
Beispiel:
Feld: 4 1 9 1 3
Entfernen des Zahl 1 ergibt:
Feld: 4 9 1 3

`int sucheZahl(int v[], int zahl)`
sucht eine Zahl im Feld und gibt ihre Position zurück.

`void sortiereFeld (int v[], int modus)`
sortiert ein Feld auf- oder absteigend (durch Wert des Parameters modus bestimmt).

`void loescheFeld(int v[])`
Lösche alle Elemente des Feldes.

`void aendereWert(int v[], int wertAlt, int wertNeu)`
Ersetzt das erste Vorkommen der Zahl wertAlt durch die neue Zahl wertneu.

`int loescheDuplikate(int v[])`
Löscht die Elemente einer Liste die mehr als 1 Mal vorkommen, so daß jedes Element genau ein Mal in der Liste vorkommt.
Beispiel:
Feld: 7 5 7 9 5 8 9
Duplikate löschen
Feld: 7 5 9 8

Bemerkungen:
B1)
In main() sollen alle obigen Funktionen ausführlich (z.B. mit Hilfe von Schleifen) getestet werden.

B2)
Weitere, selbst erfundene Funktionen implementieren.

49) Strings als Zahlen verarbeiten

Schreiben Sie Funktionen zur Zahlenverarbeitung.

`isint(string)`
prüft, ob der String eine ganze Zahl ist (d.h. aus Ziffern und evtl. Vorzeichen besteht).

`isintb(string, basis)`
prüft, ob der String eine ganze Zahl zur Basis basis ist (d.h. aus Ziffern und evtl. Vorzeichen besteht).

`isnum(string)`
prüft, ob der String eine Zahl ist (d.h. aus Ziffern und evtl. Vorzeichen bzw. Komma besteht)

`horner(zahl, basis)`
berechnet Dezimalzahl aus zahl (String) zur Basis basis.

`sumdigits(zahl)`
Berechnet die Ziffernsumme aus der gegebenen Zahl

`sumdigits_s(string)`
berechnet die Ziffernsumme aus der gegebenen Zahl (string), wenn es überhaupt eine Zahl ist. Liefert -1, wenn keine gültige Zahl.

`isipv4(string)`
prüft, ob der gegebene String eine IPv4-Adresse enthält.
IPv4-Adressen sind 4 Zahlen zwischen 0 und 255, die durch einen Punkt getrennt sind.
Beispiele: 127.0.0.1, 193.170.149.129, 81.200.64.185
Ungültige: 81.2010.64.185, 123.23.12, 127,0,0,1

`anz_vokale(string)`
zählt die Selbstlaute im String und liefere Anzahl zurück.

`anz_zahlen(string)`
ermittelt die Anzahl der ganzen Zahlen im String. Eine Zahl ist eine ununterbrochene Folge von Ziffern.
Beispiel: anz_zahlen("x 34 3 12k33 44 hallo 23") liefert 6.

`sum_zahlen(string)`
ermittelt die Summe der ganzen Zahlen im String. Eine Zahl ist eine ununterbrochene Folge von Ziffern.
Beispiel: anz_zahlen("x 34 3 12k33 44 hallo 23") liefert 149

`reverse_s(string)`
liefert String in umgekehrter Reihenfolge zurück.

`toupper(string)`
liefert String in Großbuchstaben zurück.

`tolower(string)`
liefert String in Kleinbuchstaben zurück.

`isupper(string)`
prüft, ob String nur Großbuchstaben enthält.

`islower(string)`
prüft, ob String nur Kleinbuchstaben enthält.

`ispalindrom(string)`
Prüft, ob der String ein Palindrom ist (von vorne und hinten gelesen gleich).

`ispalindrom_l(liste)`
Prüft, ob die Liste von vorne und hinten gelesen gleich ist.

`reverse_l(liste)`
liefert die Liste in umgekehrter Reihenfolge zurück.

`sum(liste)`
liefert die Summe aller Zahlen in der Liste

`iskonto(string)`
prüft, ob der String eine gültige Kontonummer enthält. Dabei gelten folgende Regeln:
R1: genau 9 Ziffern (es dürfen Leerzeichen dazwischen sein
R2: die letzte Ziffer ist eine Prüfziffer und es gilt Die Kontonummer ist genau 9 Stellen lang und ist dann richtig, wenn die folgendermaßen gebildete Summe auf Null endet:
Ziffern mit ungeradem Index werden addiert (=Teilsumme1); Ziffern mit geradem Index werden verdoppelt und das jeweilige Produkt addiert, wenn es einstellig ist, andernfalls wird die Quersumme des Produkts addiert (=Teilsumme2);
Summe= Teilsumme1 + Teilsumme2
Beispiel:
123456782 1+ 3+ 5+ 7+ 2 = 18 Teilsumme1
2*2+ 4*2+ (1+2)+ (1+6) = 22 Teilsumme2
6*2=12 8*2=16
---
40 Summe mod
10= Null

Beispiele für Kontonummern:
697199107 richtige Kontonummer
723016699 falsche Kontonummer

50) Weitere Funktionen

`rmdups(liste)`
(ReMove DUPlicates) liefert Liste, aus der alle Vielfachen entfernt (hintereinander) werden.
Beispiele:
[1, 2, 2, 2, 5, 2, 3, 2, 2, 23] ->
[1, 2, 5, 2, 3, 2, 23]
[99, 99, 23, 99, 99, 12, 2] -> [99, 23, 99, 12, 2]
[] -> []
[1, 2, 4, 10] -> [1, 2, 4, 10]"""

`rmalldups(liste)`
(ReMove ALL DUPlicates) liefert Liste, aus der alle Vielfachen entfernt (müssen nicht hintereinander stehen).
Beispiele:
[1, 2, 2, 2, 5, 2, 3, 2, 2, 23] -> [1, 2, 5, 3, 23]
[99, 99, 23, 99, 99, 12, 2] -> [99, 23, 12, 2]
[] -> []
[1, 2, 4, 10] -> [1, 2, 4, 10]

`binsearch(liste, such)`
liefert den Index von such in der sortierten Liste liste oder -1, wenn such in liste nicht vorkommt. Es ist die binäre Suche anzuwenden!

`sumdigits(zahl)`
berechnet die Ziffernsumme aus der gegebenen integer-Zahl

`sumteilbar(liste, teiler)`
Berechnet die Summe aller durch teiler teilbaren Zahlen der Liste.

`issorted(liste)`
Prüft, ob die Liste sortiert ist, d.h. ob
e1 <= e2 <= e3 <= ... <= en
e1, e2, ... en sind die n Elemente der Liste mit Länge n.

`printlxl(liste1, liste2)`
gibt eine Multiplikationstabelle der beiden Listen auf dem Bildschirm aus aus.
Zum Beispiel:
a = [1, 2, 4, 5]
b = [3, 2, 2]
printlxl(a, b) gibt dann folgendes aus:
3 2 2
6 4 4
12 8 8
15 10 10
Die erste Liste bestimmt die Anzahl der Zeilen, die zweite die Anzahl der Spalten. Es wird immer ein Element der ersten Liste mit einem der zweiten Liste multipliziert.

`lxl(liste1, liste2)`
liefert eine Multiplikationstabelle der beiden Listen (Liste von Listen).
Zum Beispiel:
a = [1, 2, 4, 5]
b = [3, 2, 2]
lxl(a, b) liefert folgende Liste zurück:
[[3, 2, 2], [6, 4, 4], [12, 8, 8], [15, 10, 10]]
übersichtlich formatiert:
[[3, 2, 2],
[6, 4, 4],
[12, 8, 8],
[15, 10, 10]]
Die erste Liste bestimmt die Anzahl der "Zeilen", die zweite die Anzahl der "Spalten". Es wird immer ein Element der ersten Liste mit einem der zweiten Liste multipliziert.

`checksums(index, array)`
prüfe, ob Summe der Zahlen in array links vom index gleich Summe rechts vom index"""

`void loescheMinMax(int v[])`
entfernt das Minimum und Maximum aus dem Feld v
Sollte das Minimum oder Maximum mehrfach vorkommen, so sind alle Minima bzw. Maxima zu entfernen.

`void addiereZahl(int v[], int zahl)`
addiert zu allen Elementen des Felds v die Zahl zahl.

`void umdrehen(int v[])`
bringt die Elemente des Felds v in die umgekehrte Reihenfolge.

`void filtern(int v[], int zahl)`
Filtert alle Zahlen aus dem Feld v, die kleiner als die Zahl zahl sind.

`int bildeSumme(int v[])`
Bildet die Summe aller Zahlen aus dem Feld v.

`void transponiere(...)`

Transponiert alle Zahlen eines zweidimensionalen intger-Felds.

D.h. es werden alle Feldelemente des zweidimensionalen feldes an der Hauptdiagonalen (von links oben nach rechts unten) gespiegelt.

Beispiel:

1	2	3		1	4	7
4	5	6	>>	2	5	8
7	8	9		3	6	9

C-ÜBUNGSAUFGABEN FUNKTIONEN-OUTPUT 4

51) Mathematische Probleme

51.1) Gründsätzliches und Philosophisches zu mathematischen Beweisen

Definition:
Eine positive ganze Zahl heißt 'von geradem Typ', wenn ihre Primfaktorzerlegung eine gerade Anzahl von Primzahlen enthält. Sonst heißt sie 'von ungeradem Typ'.

Beispiel:
4 = 2*2 ist von geradem Typ.
18 = 2*3*3 ist von ungeradem Typ.
Weil 1 das Produkt von 0 Primzahlen ist, ist die 1 somit von geradem Typ.

Definition:
E(n) bezeichnet die Anzahl der positiven ganzen Zahlen von geradem Typ, die kleiner oder gleich n sind.
O(n) bezeichnet die Anzahl der positiven ganzen Zahlen von ungeradem Typ, die kleiner oder gleich n sind.

Der Mathematiker Polya stellte 1919 folgende Behauptung auf:
Für alle $n \geq 2$ gilt: $O(n) \leq E(N)$

Die folgende Tabelle soll helfen, die Vermutung zu überprüfen:

n	Primfaktorzerlegung	Typ	E(n)	O(n)
1	-	gerade	-	-
2	2	ungerade	1	1
3	3	ungerade	1	2
4	2^2	gerade	2	2
5	5	ungerade	2	3
6	2*3	gerade	3	3
7	7	ungerade	3	4
8	2^3	ungerade	3	5
9	3^2	gerade	4	5
10	2*5	gerade	5	5
11	11	ungerade	5	6
12	2^2*3	ungerade	5	7
...				

Schreiben Sie ein Programm, das die Vermutung für "hinreichend" viele Zahlen überprüft.
Sind Sie überzeugt, dass die Vermutung korrekt ist ?

51.2) Rätsel
Ein nach Christus geborener Mathematiker wurde nach seinem Alter gefragt.
Er antwortete:
„Ich bin in einem Primzahlzwillingsjahr geboren. Multipliziere ich die Quersumme mit dem Querprodukt meines Geburtsjahres, so erhalte ich die Fakultät einer vollkommenen Zahl."
In welchem Jahr wurde der Mathematiker geboren?

51.3) Kaprekar- Konstanten
Algorithmen zur Erzeugung von Kaprekar- Konstanten
Von dem indischen Mathematiker Kaprekar stammt folgender Zusammenhang :
a) man nimmt eine beliebige 4- stellige Zahl Z , bei der nicht alle Ziffern gleich sind.
b) Aus dieser Zahl Z bildet man durch Umordnung der Ziffern eine größte Zahl Z1, und eine kleinste Zahl Z2. Dann berechnet man eine neue Zahl aus der Differenz Z1 – Z2.
c) Dieser Vorgang wird wiederholt. Nach einigen Schritten landet man stets bei der Zahl 6174. Diese Zahl reproduziert sich bei weiteren Schritten selbst.
Beispiel : Z = 4732 (beliebige Startzahl mit 4 unterschiedlichen Ziffern) :
7432 – 2347 = 5085 ; 8550 – 558 = 7992 ; 9972 – 2799 = 7173 ;
7731 – 1377 = 6354 ; 6543 – 3456 = 3087 ; 8730 – 378 = 8352 ;
8532 – 2358 = 6174 ; 7641 – 1467 = 6174 ;
Die Kaprekar- Konstante für eine 4- stellige Zahl lautet somit : 6174 ;

51.4) Giuga-Zahlen
Eine natürliche Zahl n ist eine Giuga-Zahl, wenn alle ihre Primteiler p den Wert n/p - 1 teilen.
Schreiben Sie ein Programm, das bestimmt, ob eine natürliche Zahl eine Giuga-Zahl ist.

Beispiel:
30 ist eine Guiga-Zahl.
Die Primteiler von 30 sind: 2, 3 und 5
2 teilt 30/2-1=14 und 3 teilt 30/3-1=9 und 5 teilt 30/5-1=5

Hinweis: Jede Primzahl ist eine Giuga-Zahl. Es ist derzeit (2016) nicht bekannt, ob die Umkehrung dieser Aussage gilt.

51.5) Siehe UE UE_Schleifen
a) Geben Sie die ersten 100 Primzahlen auf derm Bildschirm aus.
b) Geben Sie die ersten 100 Primzahlzwillinge auf derm Bildschirm aus.
c) Geben Sie die ersten 4 vollkommenen Zahlen auf derm Bildschirm aus.
d) Geben Sie die ersten 10 defizienten Zahlen auf derm Bildschirm aus.
e) Geben Sie die ersten 10 abundanten Zahlen auf derm Bildschirm aus.
f) Geben Sie die ersten 5 Mersenne-Primzahlen auf derm Bildschirm aus.
g) Geben Sie die ersten 100 Harshad-Zahlen auf derm Bildschirm aus.
h) Geben Sie die ersten 10 A-Zahlen auf derm Bildschirm aus.

51.6)
Von einer ganzen Zahl sollen alle Teiler in einem Feld abgespeichert werden.
Danach sollen diese auf dem Bildschirm ausgegeben werden.
Das Programm soll zusätzlich ermitteln, ob alle diese Teiler in das Feld passen (d.h. die Feldlänge ausreichend ist).

51.7)
Eine Sierpinski-Zahl (benannt nach dem polnischen Mathematiker Wacław Sierpiński) ist eine ganze, ungerade Zahl k > 0, für die die unendliche Zahlenfolge $k \cdot 2^n + 1$ mit $n \geq 1$ keine Primzahlen enthält.
Beispiel:
k = 3 ist keine Sierpinski-Zahl, da die Folge
$3 \cdot 2^1 + 1, 3 \cdot 2^2 + 1, 3 \cdot 2^3 + 1, 3 \cdot 2^4 + 1, ...$
ausgerechnet:
7, 13, 25, 49, ...
(mindestens) eine Primzahl enthält.
Zeigen Sie, dass 19 keine Sierpinski-Zahl ist.

51.8) Quersummen
a) Schreiben Sie eine Funktion, die die Quersumme einer Zahl berechnet.

b) Schreiben Sie eine Funktion, die alle Zahlen innerhalb eines vorgegebenen Zahlenbereichs (z.B. zwischen 10 und 1000) berechnet, die eine bestimmte, vorgegebene Quersumme haben.

c) Schreiben Sie eine Funktion, die alle Zahlen innerhalb eines vorgegebenen Zahlenbereichs berechnet, die das Vielfache einer bestimmten, vorgegebene Quersumme sind.

d) Schreiben Sie eine Funktion, die berechnet, welche Quersumme innerhalb eines bestimmten vorgegebenen Zahlenbereichs am häfigsten vorkommt.
(Beispiel: Welche Quersumme kommt bei allen Zahlen zwischen 100 und 1000 am häufigsten vor?)

e) Wenn von einer Quersumme so lange wieder die Quersumme gebildet wird, bekommt man die sogenannte "Einstellige Quersumme".
Schreiben Sie eine Funktion, die die "Einstellige Quersumme" einer Zahl berechnet.
Beispiel: 987 --> 9+8+7=24 --> 2+4 = 6

f) Schreiben Sie eine Funktion, die das Querprodukt einer Zahl berechnet.
Beispiel: 237 --> 2*3*7=42

g) Eine Zahl, bei der die Summe aus Quersumme und Querprodukt wieder die Zahl selbst ergibt, nennt man hier Q-Zahl.
Beispiel: 79 = 7 + 9 + 7*9 = 79.
Schreiben Sie eine Funktion, die alle Q-Zahlen innerhalb eines vorgegebenen Zahlenbereichs berechnet (z.B. zwischen 0 und 1000).

51.9) Die Zahl 100
Kann man alle natürlichen Zahlen bis 100 als Summe dreier Kubikzahlen darstellen?
Als Gleichung ausgedrückt:
$x^3 + y^3 + z^3 = n$
Probieren Sie mit brute force.
Wie beantworten Sie die Frage ?

Bemerkungen zu Vermutungen mit großen Zahlen:
siehe dazu:
http://www.math.uni-bremen.de/didaktik/ma/ralbers/Materialien/Vortragsmat/GrosseZahlen.pdf

1) Vermutung von Euler:
1769 vernutete der Mathematiker Leonard Euler, daß die Gleichung
$x^4 + y^4 + z^4 = w^4$
keine Lösung mit nur natürlichen Zahlen hat.
Gegenbeispiel:
$18.796.760^4 + 15.365.639^4 + 2.682.440^4 = 20.615.673^4$
Es gibt sogar unendlich viele Lösungen.

2) Vermutung von Polyha (siehe oben)
1980 beweist der japanische Mathematiker Minoru Tanaka, dass
906150257 die kleinste Obergrenze ist, für die die Polya-Vermutung falsch ist.

3) Vermutung über die Zahl 100
Spektrum der Wissenschaft 11.19 Seite 8
Seit 1954 interessieren sich Mathematiker für folgendes Problem:
Kann man alle natürlichen Zahlen bis 100 als Summe dreier Kubikzahlen darstellen?
Als Gleichung ausgedrückt:
$x^3 + y^3 + z^3 = n$
Die Summe 42 erwies sich als sehr schwierig. Im jahr 2019 wurde das Problem gelöst:
$(-80538738812075974)^3 + 80435758145817515^3 + 12602123297335631^3 = 42$

C-ÜBUNGSAUFGABEN FUNKTIONEN-OUTPUT 5

52) Sportwetten

Sportwetten
Am 28.6.2012 waren die Quoten in der Fuball-Europameisterschaft für das Spiel Deutschland - Italien wie folgt:

Sieg Deutschland:	1,9
Unentschieden:	3,35
Niederlage Deutschland:	4,7

Das bedeutet:
Wenn man z.B.auf eine Niederlage Deutschlands 100 Euro gesetzt hat, bekommt man
$4{,}7 \cdot 100 = 470$ Euro Geld vom Wettbüro mybet.com
Bei einem Einsatz von 100 Euro hat man also einen Gewinn von
470 Euro - 100 Euro = 370 Euro
gemacht.

II) Spielstrategien
Kann man eine Wette so gestalten, daß man auf jeden Fall einen Gewinn macht?
Nein, da der Mensch keine "göttlichen Fähigkeiten" der Prognose besitzt, kann er eine Wette verlieren und damit hat man den Einsatz verloren.
Aber man kann mehrere Wetten abschließen und damit eine **Gesamtwette** bilden, die aus mehreren **Teilwetten** besteht.
Kann man eine Gesamtwette so gestalten, dass man auf jeden Fall einen Gesamtgewinn macht?
Ja, falls die Quoten es zulassen (siehe folgendes Besipiel)

Gewinnbringende Gesamtwette

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
10	10	10
100	200	300
= 10*100-(100+200+300) = 400	= 10*200-(100+200+300) = 1400	= 10*300-(100+200+300) = 2400

Minimaler Gesamtgewinn = 2400

Definition:
Eine Teilwette eines Gesamtwette heißt **potentiell gewinnbringend** genau dann wenn bei einem Gewinn dieser Wette der Geamtgewinn > 0 ist.
Man muß also die Einsätze so bilden, daß alle 3 Wetten jeweils potentiell gewinnbringend sind.
Da das Wettbüro die Quoten so berechnet, daß dies nie der Fall sein wird (sonst wäre das Wettbüro pleite), wird die Strategie gewählt, die eine möglichst große Anzahl (also 2) von Teilwetten jeweils potentiell gewinnbringend macht.
Unter diesen wählt man dann die mit dem größten Gewinn aus.

III) Meine Spielstrategie
Bei einem vorgegebenen maximalem Verlust, den die Gesamtwette nicht überschreiten darf, werden 2 potentiell gewinnbringende Teilwetten gesucht, so daß deren minimaler potentiell gewinnbringender Gesamtgewinn möglichst groß wird.

Beispiel

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	5
50	200	100
= 2*50-(50+200+100) = -250	= 3*200-(50+200+100) = 250	= 5*100-(50+200+100) = 150

Minimaler Gesamtgewinn = -250
Minimaler potentiell gewinnbringender Gesamtgewinn = 150

Wichtige Frage:
Kann man eine Gesamtwette finden, die einen minimaler Gesamtgewinn nicht unterschreitet (also garantiert, daß ein möglicher Verlust nicht überschritten wird) und der minimale potentielle gewinnbringende Gesamtgewinn möglichst groß werden läßt.
Dies könnte man -sofern man genügend Zeit hat - durch "Probieren" herausfinden.

IV) Konkrete Aufgabe
Gesucht: Eine Gesamtwette mit
Minimalem Gesamtgewinn = -20 und minimalem potentiell gewinnbringendem Gesamtgewinn.

Suche durch Probieren

Wette 1

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
4	7	9
= 2*4-(4+7+9) = -12	= 3*7-(4+7+9) = 1	= 4*9-(4+7+9) = 16

Minimaler Gesamtgewinn = -12
Minimaler potentiell gewinnbringender Gesamtgewinn = 1

Wette 2

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
4	10	11
= 2*4-(4+10+11) = -17	= 3*10-(4+10+11) = 5	= 4*11-(4+10+11) = 19

Minimaler Gesamtgewinn = -17
Minimaler potentiell gewinnbringender Gesamtgewinn = 5

Wette 3

Quote q1 Einsatz E1 Gewinn G1	Quote q2 Einsatz E2 Gewinn G2	Quote q3 Einsatz E3 Gewinn G3
2	3	4
0	11	8
= 2*0-(0+11+8) = -19	= 3*11-(0+11+8) = 14	= 4*8-(0+11+8) = 13

Minimaler Gesamtgewinn = -19
Minimaler potentiell gewinnbringender Gesamtgewinn = 13

Bis jetzt ist Wette 3 am besten.

Aufgabe:
Schreiben Sie ein Programm, das durch "Brute Force" eine optimale Wette bestimmt.

s

C-ÜBUNGSAUFGABEN FUNKTIONEN-OUTPUT 6

Bemerkung:
Diese weiter unten dokumentierten Funktionen werden später in komplexeren Programmen benötigt.
Des wegen müssen sie unbedingt implemetiert werden.

53) Magisches Quadrat

Beschreibung des Rätsels:
Ein teilweise (oder völlig) unbelegtes 3x3-Feld (kurz Feld bzw. Neunerfeld) soll so mit verschiedenen Ziffern von 1 bis 9 besetzt (magisch ergänzt) werden, dass es magisch wird, d.h alle Ziffern müssen verschieden und die jeweiligen Zeilensummen, Spaltensummen, Hauptdiagonalensummen müssen gleich groß sein.

Beispiel:
Unbelegte Zellen werden mit 0 gekennzeichnet.

0	0	0
1	2	3
0	0	0

Dieses Feld ist nicht magisch ergänzbar

0	4	0
9	5	3
0	3	0

Dieses Feld ist nicht korrekt

7	12	0
9	11	0
0	8	0

Dieses Feld ist nicht korrekt

Bemerkung:
Programmtechnisch wird dieses zweidimensionale 3 x 3-Feld durch ein eindimensionales Feld der Länge 9 dargestellt:

1	2	3
4	5	6
7	8	9

----------->

1	2	3	4	5	6	7	8	9

```
/************************************************************/
/**                                                        **/
/**  void kopieren(const int quelle[],int ziel[])          **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) const int quelle[] : das zu kopierende Neunerfeld
  (o) int ziel[] : das kopierte Neunerfeld

Return:
  Kein

Beschreibung:
  Kopiert das Neunerfeld "quelle" in das Neunerfeld "ziel".
  Ein Neunerfeld ist ein eindimensionales Feld der Länge 9.
*/


/************************************************************/
/**                                                        **/
/**  void kopieren(int ziel [], int wert)                  **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (o) int ziel[] : das kopierte Neunerfeld
  (i) int wert : den zu kopierenden Wert

Return:
  Kein

Beschreibung:
  Kopiert in jede Zelle des Neunerfeldes "ziel" die Zahl "wert"
*/


/************************************************************/
/**                                                        **/
/**  void kopieren(int ziel [],int z0, int z1, int z2, int z3, **/
/**                int z4, int z5, int z6, int z7, int z8)     **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (o) int ziel[] : das kopierte Neunerfeld
  (i) int z0 : die 1. zu kopierende Zahl
      ...
  (i) int z8 : die 8. zu kopierende Zahl

Return:
  Kein

Beschreibung:
  Kopiert die Zahlen z0, ..., z8 in das Neunerfeld "ziel"
  also: ziel[0]=z0, ..., ziel[8]=z8
*/
```

```
/*************************************************************/
/**                                                         **/
/**  bool sindZahlenKorrekt(const int v[])                  **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  true : in v kommen keine 2 gleichen Zahlen vor.
  false: in v kommen 2 gleichen Zahlen vor.


Beschreibung:
  Prüft nach, ob alle Zahlen der belegten Zellen im "feld" v
  (eine belegte Zelle hat einen Wert != 0) korrekt sind, d.h.
  nicht mehrfach vorkommen.
*/


/*************************************************************/
/**                                                         **/
/**  bool sindZahlenKorrekt(const int v[], int pos, int wert) **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld
  (i) int pos 0<=pos<=8 : Position im Neunerfeld v
  (i) int wert 1<=wert<=9 : Wert von v an der Stelle pos

Return:
  true : v könnte an der Stelle pos auf "wert" gesetzt werden.
  false: v könnte an der Stelle pos nicht auf "wert" gesetzt
         werden, da sonst 2 gleiche Zahlen in v vorkommen.


Beschreibung:
  Prüft nach, ob alle Zahlen der belegten Zellen im "feld" v
  (eine belegte Zelle hat einen Wert != 0) korrekt wären
  (also nicht mehrfach vorkommen), wenn im Neunerfeld v die
  durch pos bezeichnete Stelle auf "wert" gesetzt würde.
*/
```

```
/************************************************************/
/**                                                        **/
/**  bool sindSummenKorrekt(const int v[])                 **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  true : alle Zeilen-, Spalten- und Diagonalensummen sind gleich.
  false: sonst

Beschreibung:
  prueft, ob die Zeilen-, Spalten- und Diagonalensummen aller
  3-Reihen, die keine unbelegten Zellen enthalten, gleich sind.
*/


/************************************************************/
/**                                                        **/
/**  bool sindSummenKorrekt(int v[], int pos, int wert)    **/
/**                                                        **/
/*#***********************************************************/
/*
/*
Parameter:
  (i) const int v[] : Neunerfeld
  (i) int pos 0<=pos<=8 : Position im "feld" v
  (i) int wert 1<=wert<=9 : Wert in v an der Stelle pos

Return:
  true : alle Zeilen-, Spalten- und Diagonalensummen sind gleich.
  false: sonst

Beschreibung:
  prueft, ob die Zeilen-, Spalten- und Diagonalensummen aller voll
  belegten 3-Reihen im "feld" v gleich sind, wenn man im "feld" v
  die durch pos bezeichnete Stelle auf "wert" gesetzt hätte.
*/

/************************************************************/
/**                                                        **/
/**  bool korrekt(const int v[])                           **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  true : das "feld" v ist korrekt
  false: sonst

Beschreibung:
  prueft, ob das das "feld" v korrekt ist, also keine 2 gleichen
  Zahlen (auf belegte Zellen bezogen) enthält und ob die Zeilen-,
  Spalten- und Diagonalensummen aller 3-Reihen, die keine
  unbelegten Zellen enthalten, gleich sind.
*/
```

```
/************************************************************/
/**                                                        **/
/**  bool totalkorrekt(const int v[])                      **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  true : das "feld" v ist total korrekt
  false: sonst

Beschreibung:
  prueft, ob das das "feld" v korrekt ist, also keine 2 gleichen
  Zahlen (auf belegte Zellen bezogen) enthält und ob die Zeilen-,
  Spalten- und Diagonalensummen aller 3-Reihen, die keine unbelegten
  Zellen enthalten, gleich sind.
  Außerdem wird noch geprüft, ob alle Zahlen im "feld" v zwischen
  1 und 9 sind (je einschließlich).
  Diese Funktion wird z.B. nur dann benutzt, um ein vom Anwender
  erstellten Neunerfeld auf totale Korrektheit zu überprüfen.
  Ein in der Funktion erstelleNachfolger(...) erzeugter Nachfolger
  muß hingegen nicht auf totale Korrektheit überprüft werden, da
  dieser Nachfolger nicht vom Anwender erzeugt (und deshalb)
  fehlerhaft sein kann.
*/

/************************************************************/
/**                                                        **/
/**  bool korrekt(int v[], int pos, int wert)              **/
/**                                                        **/
/*#***********************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld
  (i) int pos 0<=pos<=8 : Position im "feld" v
  (i) int wert 1<=wert<=9 : Wert in v an der Stelle pos

Return:
  true : das "feld" v ist korrekt
  false: sonst

Beschreibung:
  prueft, ob das das "feld" v korrekt ist, wenn man im "feld" v
  die durch pos bezeichnete Stelle auf "wert" gesetzt hätte.
*/
```

```
/*************************************************************/
/**                                                         **/
/**  bool istVollbelegt(const int v[])                      **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  true : das "feld" v ist voll belegt
  false: sonst

Beschreibung:
  Eine Zelle im "feld" v ist unbelegt, wenn sie den Wert 0 hat.
  Sie ist belegt, wenn sie als Wert 1, 2, 3, 4, 5, 6, 7, 8 oder 9
hat.
  Es wird geprüft, ob jede Zelle des Feldes mit einer ganzen Zahl
zwischen
  1 und 9 belegt ist.
*/


/*************************************************************/
/**                                                         **/
/**  void druckeFeld(const int v[])                         **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld

Return:
  Kein

Beschreibung:
  Gibt "feld" v auf dem Bildschirm aus.
*/
```

```
/*************************************************************/
/**                                                         **/
/**  bool erstelleNachfolger(int v[], int pos,              **/
/**                          int wert, int nachfolger[])    **/
/**                                                         **/
/*#************************************************************/
/*
Parameter:
  (i) const int v[] : Neunerfeld
  (i) int pos 0<=pos<=8 : Position im "feld" v
  (i) int wert 1<=wert<=9 : Wert von v an der Stelle pos
  (o) int nachfolger : Nachfolger von v

Return:
  true : Es gibt einen Nachfolger von v
  false: Es gibt keinen Nachfolger von v

Beschreibung:
  Prueft nach, ob das das "feld" v , wenn man es an der Stelle
  pos auf "wert" gesetzt hätte, ein Nachfolger wäre, d.h.
  korekt wäre.
  Wenn dies der Fall wäre, wird dieser Nachfolger im Neunerfeld
  "nachfolger" zurükgeliefert.
  Wenn es keinen Nachfolger gibt, wird "nachfolger" auf null
  gesetzt und false zurückgeliefert.
*/
```

54) Aufgabe (Rätsel der Woche 26.4.20)

Zwei Mathematikerinnen begegnen sich zufällig auf der Straße.
"Wie alt sind eigentlich deine vier Kinder inzwischen?", fragt die erste.
"Die Summe der Alter aller 4 Kinder ergibt 15 Jahre, und das Produkt ergibt die Hausnummer hinter dir", antwortet die zweite.
Die erste Mathematikerin überlegt eine Weile und sagt dann:
"Das reicht mir nicht, da brauche ich doch noch mehr Infos."
Darauf sagt die zweite Frau: "Also Zwillinge hab ich ja keine."
"Jetzt weiß ich Bescheid", entgegnet die erste.
Wie alt sind die vier Kinder der zweiten Mathematikerin?
Hinweis: Das Alter wird in ganzen Zahlen angegeben.

Hinweise zur programmtechnischen Lösung:
1) Erstellen Sie ein 2-dimensionales Feld mit Zeilenlänge 5.
In diesem Feld werden alle mögliche Alterskombination (der 4 Kinder) mit Gesamtalter 15 und dem Gesamtaltersprodukt der 4 Kinder gespeichert, wie z.B:
1,2,3,4,24
3,4,3,5,180
...
1,2,3,4 sind die Altersangaben der 4 Kinder und 24 ist das Produkt der 4 Altersangaben.

2) Löschen Sie in diesem Feld alle mehrfachen Alterskombinationen heraus, wie z.B:
1,2,3,4,24
4,3,2,1,24
4,3,1,2,24
usw.

3) Sortieren Sie die verbleibende Liste nach dem Produkt der Alterskombinationen