Aufgabe 1: Ein Algorithmus in Form eines *Flussdiagrammes* oder *Struktogrammes* soll die Umrechnung von **Fahrenheit-**Werten in **Celsius**-Werte ermöglichen. Die Variablen *lower*, *upper* und *step* sollen eingelesen werden. Diese Variablen beschreiben in der genannten Reihenfolge die untere und obere Fahrenheit-Temperatur und *step* die Schrittweite ($0 < step$ gefordert). Die Umrechnungsformel lautet: $celsius = 5.0 * (fahrenheit - 32.0) / 9.0$.
Von *lower* bis *upper* sollen mit der Schrittweite *step* zu jedem Fahrenheit-Wert der zugehörige Celsius-Wert berechnet und ausgegeben werden. Der Algorithmus kann optional als C-Programm implementiert werden.

Aufgabe 2: Gegeben sind die drei Koeffizienten *a, b, c,* wobei diese beliebige reele Werte haben können. Gesucht sind alle reellen und komplexen Werte von *x*, für welche die Gleichung $\mathbf{a * x^2 + b * x + c = 0}$ erfüllt ist. Im Falle von $\sqrt{-1}$ setzen Sie den Buchstaben ***i*** ein.
Der Algorithmus kann optional als C-Programm implementiert werden.

Aufgabe 3: Gegeben seien 2 Zeitspannen der Form *tag; h; min; sec; msec* . Formulieren Sie ein Flußdiagramm, welches die **Summe** der beiden Zeitspannen berechnet und von der Summe die *tag-*, *h-, min-, sec-* und *msec*-Werte ausgibt. Für jeden Summanden sind die *tag-*, *h-, min-, sec-* und *msec*-Werte am Anfang einzulesen. Für die Werte sind folgende Intervalle einzuhalten: $0 <= tag$, $0 <= h$, 0<= min, 0<= sec, 0<= msec . Bei der Eingabe wird solange eine Schleife für jeden Wert durchlaufen, bis der gelesene Wert die angegebene Bedingung erfüllt. Der Algorithmus kann optional als C-Programm implementiert werden.

Aufgabe 4: In unserem Kalender sind zum Ausgleich der astronomischen und kalendarischen Jahreslänge in regelmäßigen Abständen Schaltjahre eingebaut. Zur exakten Festlegung der Schaltjahre dienen die folgenden Regeln:
i. Ist die Jahreszahl durch 4 teilbar, so ist das Jahr ein Schaltjahr. Diese Regel hat allerdings eine Ausnahme:
ii. Ist die Jahreszahl durch 100 teilbar, so ist das Jahr kein Schaltjahr. Diese Ausnahme hat wiederum eine Ausnahme:
iii. Ist die Jahreszahl durch 400 teilbar, so ist das Jahr doch ein Schaltjahr.

Erstellen Sie ein Flußdiagramm, das berechnet, ob eine vom Benutzer eingegebene Jahreszahl ein Schaltjahr ist oder nicht. Der Algorithmus kann optional als C-Programm implementiert werden.

Aufgabe 5: Es ist der Funktionswert $y = \sqrt{a}$ für a = 4 mit dem sumerisch-babylonischen Wurzelziehen (**Heron-Verfahren**) **manuell** iterativ für 4 Iterationsschritte (x_1, y_1) bis (x_4, y_4) zu berechnen, wobei mit dem Iterationsverfahren
$\mathbf{x_1 := (x_0 + y_0) / 2}$, $\mathbf{y_1 := a / x_1}$ bzw. $\mathbf{x_{n+1} := (x_n + y_n) / 2}$, $\mathbf{y_{n+1} := a / x_{n+1}}$
gearbeitet werden soll. Für jedes i (1 <= i <= 4) ist $\mathbf{x_i * y_i}$ auszurechnen und zu notieren.
Alternativ ist $\mathbf{x_4}$ mittels der folgenden Formel auszurechnen und zu notieren:
$\mathbf{x_0 := a / 2}$ und $\mathbf{x_i := (x_{i-1} + a / x_{i-1}) / 2}$, $\mathbf{i = 1, 2, 3, ...}$

Aufgabe 6: Der größte gemeinsame Teiler ist mittels des ***Euklidischen Algorithmus*** sowohl über **ggT(54, 81)** *manuell rekursiv* als auch über **Euklid(54, 81)** *manuell iterativ* zu bestimmen:

```
long ggT(long n, long m){                    // Rekursiver Algorithmus von Euklid
      if(abs(m)>abs(n)) return abs(ggT(m,n)); // vertauschen von n und m
      if(!m) return abs(n);                  // if(m == 0) return abs(n), Abbruch
      return abs(ggT(m,n%m));                // rekursiver Schritt
}
```

```
long Euklid(long a, long b){                    // Iterativer Algorithmus von Euklid
      a = abs(a); b=abs(b);
      while(a*b){
                  if(a >= b) a = a % b;
                  else b = b % a;
     }
     return a + b;
}
```

Aufgabe 7: Die **Fakultät** einer natürlichen Zahl ist wie folgt definiert: 0! = 1 , 1! = 1, n! = (n-1)! * n , für n >= 1. Formulieren Sie sowohl ein Struktogramm für die **rekursive** als auch für die **iterative** Berechnung. Die Algorithmen können optional als C-Programm implementiert werden (rekursiv und iterativ).

Aufgabe 8: Fibonacci - Zahlen F(n) werden für natürliche Zahlen n >= 2 **rekursiv** über die Formel **F(n) := F(n-1) + F(n-2)** berechnet, wobei die Anfangswerte **F(0) = 0** für n = 0, **F(1) = 1** für n = 1 gelten. Alternativ **analytisch** erfolgt die Berechnung von **F(n)** über die folgende Formel:

$$F(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Formulieren Sie ein Struktogramm für die **rekursive**, die **analytische** und die **iterative** Berechnung der Fibonacci-Zahlen. Die Algorithmen können optional als C-Programme implementiert werden (jeweils rekursiv, analytisch und iterativ).

Aufgabe 9: Die **Ackermannsche Funktion** ist ein Beispiel für eine **totale** und **berechenbare** Funktion, die **nicht primitiv-rekursiv** ist. Diese Funktion ist **mehrfach rekursiv**:

Ackermann(m, n) := wenn m == 0, dann n + 1;
wenn n == 0, dann Ackermann(m - 1, 1)
sonst Ackermann(m-1, Ackermann(m, n-1))

a.) Rechnen Sie **Ackermann(2, 3)** manuell aus.
b.) Formulieren Sie ein Struktogramm zur Berechnung der Ackermannschen Funktion.
Der rekursive Algorithmus kann optional als C-Programm implementiert werden.

Aufgabe 10:
Binomialkoeffizienten $\binom{n}{k}$ beschreiben die Anzahl der Möglichkeiten, **k** Elemente aus einer

Grundgesamtheit **n** auszuwählen, wobei immer **0 <= k <= n** gelten muß und **n** und **k** natürliche Zahlen sind. Ein <u>numerisch stabiles</u> Verfahren, um Binomialkoeffizienten zu berechnen, ergibt sich aus folgender Formel: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$

Es gilt $\binom{n}{0} = 1$ und $\binom{n}{1} = n$ und $\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$

Der Algorithmus ist **rekursiv** und **iterativ** zu entwerfen. Beide Algorithmen können optional als C-Programme implementiert werden.

Aufgabe 11: Eine weitere mehrfach rekursive Funktion, ähnlich der Ackermann-Funktion, ist die wie folgt definierte **Hofstadter-Funktion**:

hof(n) := 1, wenn n<= 2
hof(n) := hof(n - hof(n-1)) + hof(n - hof(n-2)), wenn n>2

a.) Rechnen Sie **hof(5)** manuell aus.
b.) Formulieren Sie je einen Programmablaufplan für die rekursive und für die iterative Berechnung, implementieren Sie optional je ein C-Programm für die rekursive und die iterative Berechnung.