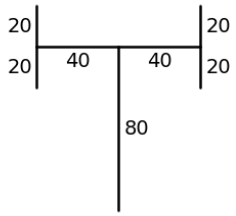
1. (3 Punkte) Skizziere das Bild, das das folgende Programm zeichnet. Schreibe die Länge an jeden Strich. Die Turtle startet in der Mitte und schaut nach rechts.

```
import turtle as t

def zeichne(laenge, bremse):
    if bremse == 0:
        return
    t.forward(laenge)
    pos = t.position()
    winkel = t.heading()
    t.left(90)
    zeichne(laenge/2, bremse-1)
    t.penup()
    t.setpos(pos)
    t.setheading(winkel)
    t.pendown()
    t.right(90)
    zeichne(laenge/2, bremse-1)

t.left(90)
zeichne(80, 3)
```

Lösung:



Zuerst ein Strich der Länge 80 nach oben. Von seinem Ende geht je ein Strich der Länge 40 nach links und nach rechts. Von deren Enden gehen je ein Strich der Länge 20 nach oben und nach unten.

2. (3 Punkte) Die folgende Funktion zeichnet eine Verzweigung.

```
def zeichne(laenge, bremse):
    if bremse == 0:
        return
    pos = t.position()
    winkel = t.heading()
    t.right(45)
    t.forward(laenge)
    zeichne(laenge*0.5, bremse-1)
    t.penup()
    t.setpos(pos)
    t.setheading(winkel)
    t.pendown()
    t.left(45)
    t.forward(laenge)
    zeichne(laenge*0.5, bremse-1)
```

   a) Gib an, wie oft `zeichne` insgesamt aufgerufen wird, wenn man `zeichne(200, 3)` aufruft. Der erste Aufruf zählt mit.

   b) Gib an, wie viele Striche bei `zeichne(200, 3)` gezeichnet werden.

   c) Gib an, wie lang die kürzesten Striche bei `zeichne(200, 6)` sind.

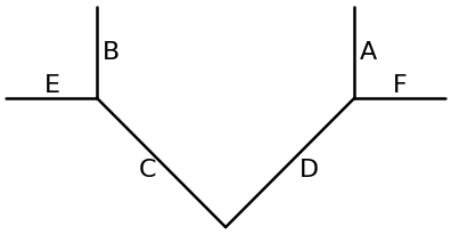
Lösung:

a) Jeder Aufruf mit `bremse` > 0 startet zwei weitere Aufrufe: $1+2+4+8 = 15$ Aufrufe. Die 8 Aufrufe mit `bremse` $= 0$ zeichnen nichts.

b) Jeder Aufruf mit `bremse` > 0 zeichnet 2 Striche. Das sind $1+2+4 = 7$ Aufrufe, also 14 Striche.

c) Der erste Aufruf zeichnet Striche der Länge 200, jede weitere Stufe halbiert die Länge. Die kürzesten Striche werden beim Aufruf mit `bremse` $= 1$ gezeichnet, also nach 5 Halbierungen: $200 : 2^5 = 6{,}25$.

3. (2 Punkte) Das Bild zeigt die Verzweigung, die die Funktion `zeichne` aus der vorigen Aufgabe mit `bremse` $= 2$ zeichnet. Die Turtle startet unten in der Mitte und schaut nach oben.



Gib die Reihenfolge an, in der die Striche A bis F gezeichnet werden.

Lösung: D, F, A, C, B, E

Zuerst wird der rechte Ast D gezeichnet. Der rekursive Aufruf an seinem Ende zeichnet erst den rechten Ast F, dann den linken Ast A. Erst danach geht es zurück nach unten zum linken Ast C, an dem dann B und E gezeichnet werden.

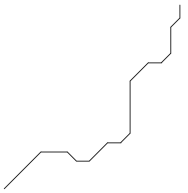
4. (3 Punkte) Betrachte noch einmal die Funktion `zeichne` für die Verzweigung.

   a) Man löscht die vier Zeilen `t.penup()`, `t.setpos(pos)`, `t.setheading(winkel)` und `t.pendown()`. Beschreibe, was dann beim Zeichnen passiert, und begründe.

   b) Man ersetzt in beiden rekursiven Aufrufen `bremse-1` durch `bremse`. Beschreibe, was dann passiert.

Lösung:

a) Der linke Ast beginnt nicht mehr am unteren Punkt des V, sondern dort, wo der rekursive Aufruf für den rechten Ast aufgehört hat. Auch die Richtung ist dann eine andere. Statt einer Verzweigung entsteht eine Zickzack-Linie:



b) `bremse` wird nie kleiner, der Basisfall `bremse == 0` wird also nie erreicht. Die Funktion ruft sich immer weiter selbst auf, bis Python mit einem `RecursionError` abbricht.

5. (4 Punkte) Die Funktion soll einen Baum zeichnen: Am Ende jedes Asts gehen zwei kürzere Äste ab, einer um 30 Grad nach rechts und einer um 30 Grad nach links gedreht. Am Ende der kürzesten Äste soll ein grüner Punkt gezeichnet werden. Ergänze die vier Lücken.

```
def zeichne(laenge, bremse):
    if bremse == 0:
        ____________________    # (1)
        return
    pos = t.position()
    ____________________        # (2)
    t.right(30)
    t.forward(laenge)
    ____________________        # (3)
    t.penup()
    t.setpos(pos)
    t.setheading(winkel)
    t.pendown()
    t.left(30)
    t.forward(laenge)
    ____________________        # (4)
```

Lösung:

```
t.dot(10, 'green')              # (1)
winkel = t.heading()            # (2)
zeichne(laenge*0.6, bremse-1)   # (3)
zeichne(laenge*0.6, bremse-1)   # (4)
```

Bei (3) und (4) ist jeder Faktor kleiner als 1 richtig, z.B. auch `laenge/2`.

6. (4 Punkte) Die Funktion zeichnet ein Sierpinski-Dreieck. Sie beginnt an der linken unteren Ecke, die Turtle schaut nach rechts.

```
def zeichne(laenge, bremse):
    if bremse == 0:
        t.pendown()
        for i in range(3):
            t.forward(laenge)
            t.left(120)
        t.penup()
        return
    halb = laenge/2
    zeichne(halb, bremse-1)
    t.forward(halb)
    zeichne(halb, bremse-1)
    t.left(120)
    t.forward(halb)
    t.right(120)
    zeichne(halb, bremse-1)
    t.right(120)
    t.forward(halb)
    t.left(120)
```

a) Gib an, wie viele Dreiecke bei `zeichne(640, 4)` gezeichnet werden und welche Seitenlänge sie haben.

b) Erkläre, warum die letzten drei Zeilen der Funktion nötig sind.

Lösung:

a) Jede Stufe teilt ein Dreieck in 3 Dreiecke mit halber Seitenlänge. Gezeichnet wird nur im Basisfall: $3^4 = 81$ Dreiecke mit der Seitenlänge $640 : 2^4 = 40$.

b) Nach dem dritten rekursiven Aufruf steht die Turtle an der linken unteren Ecke des oberen Dreiecks. Die drei Zeilen bringen sie zurück an die Ausgangsecke und drehen sie wieder nach rechts. Das ist nötig, weil sich der Aufrufer darauf verlässt: Nach dem ersten Aufruf geht er z.B. mit `t.forward(halb)` zur linken unteren Ecke des zweiten Dreiecks. Das klappt nur, wenn die Turtle nach dem Aufruf wieder an der Ausgangsecke steht und nach rechts schaut.

7. (4 Punkte) Die Funktion zeichnet eine Seite der Kochschen Schneeflocke. Die Turtle schaut am Anfang nach rechts.

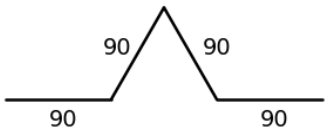
```
def zeichne(laenge, bremse):
    if bremse == 0:
        t.pendown()
        t.forward(laenge)
        t.penup()
        return
    drittel = laenge/3
    zeichne(drittel, bremse-1)
    t.left(60)
    zeichne(drittel, bremse-1)
    t.right(120)
    zeichne(drittel, bremse-1)
    t.left(60)
    zeichne(drittel, bremse-1)
```

a) Skizziere, was `zeichne(270, 1)` zeichnet, und gib die Länge der Strecken an.

b) Gib an, aus wie vielen Strecken die Linie bei `zeichne(270, 3)` besteht, wie lang jede Strecke ist und wie lang die ganze Linie ist.

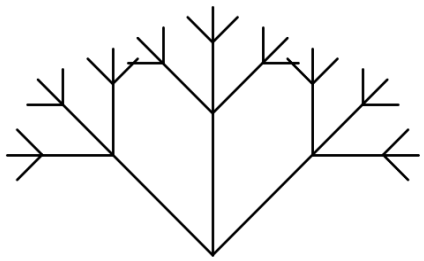
Lösung:

a) Vier Strecken der Länge 90, in der Mitte eine Spitze:



b) Jede Stufe ersetzt eine Strecke durch 4 Strecken mit einem Drittel der Länge: $4^3 = 64$ Strecken der Länge $270 : 3^3 = 10$. Die ganze Linie ist $64 \cdot 10 = 640$ lang.

8. (5 Punkte) Schreibe eine rekursive Funktion `zeichne(laenge, bremse)`, die eine dreifache Verzweigung zeichnet. Von jedem Verzweigungspunkt gehen drei Äste der Länge `laenge` ab: einer um 45 Grad nach rechts gedreht, einer geradeaus und einer um 45 Grad nach links gedreht. Am Ende jedes Asts beginnt dieselbe Verzweigung mit halber Länge. Das Bild zeigt das Ergebnis von `zeichne(160, 3)`, wenn die Turtle am Anfang nach oben schaut.



Lösung:

```
def zeichne(laenge, bremse):
    if bremse == 0:
        return
    pos = t.position()
    winkel = t.heading()
    for drehung in [-45, 0, 45]:    # rechts, geradeaus, links
        t.penup()
        t.setpos(pos)
        t.setheading(winkel)
        t.pendown()
        t.left(drehung)
        t.forward(laenge)
        zeichne(laenge/2, bremse-1)
```

Genauso richtig: die drei Äste ohne Schleife nacheinander zeichnen und vor dem zweiten und dritten Ast Position und Richtung zurücksetzen.