1. (3 Punkte) Übersetze den folgenden Pseudocode in eine Python-Funktion.

```
algorithmus anzahl_gerade(a):
    z = 0
    für jedes x in a:
        wenn x gerade:
            z = z + 1
    gib z zurück
```

Lösung:

```
def anzahl_gerade(a):
    z = 0
    for x in a:
        if x % 2 == 0:
            z = z + 1
    return z
```

2. (4 Punkte) Gegeben ist der folgende Algorithmus in Pseudocode.

```
n = 13
s = 0
solange n > 0:
    s = s + n % 2
    n = n // 2
gib s aus
```

(a) Erstelle ein Ablaufprotokoll für n und s. Was wird ausgegeben?

(b) Was berechnet der Algorithmus für eine beliebige positive ganze Zahl n?

Lösung:

Durchlauf	Start	1	2	3	4
n	13	6	3	1	0
s	0	1	1	2	3

Ausgabe: 3. Der Algorithmus zählt die Einsen in der Dualdarstellung von n.

3. (3 Punkte) Schreibe die folgende Funktion als Pseudocode.

```
def suche(a, x):
    for i in range(len(a)):
        if a[i] == x:
            return i
    return -1
```

Lösung:

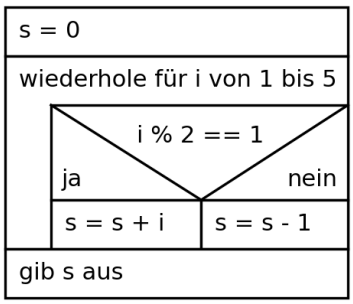
```
algorithmus suche(a, x):
    für i von 0 bis len(a)-1:
        wenn a[i] == x:
            gib i zurück
    gib -1 zurück
```

4. (3 Punkte) Gib für jede Zählschleife den passenden Schleifenkopf in Python an.
   (a) `für i von 1 bis 10:`
   (b) `für i von 0 bis n-1:`
   (c) `für i von n bis 1 abwärts:`

Lösung:

```
for i in range(1, 11):
for i in range(n):
for i in range(n, 0, -1):
```

5. (3 Punkte) Erstelle für den folgenden Algorithmus ein Ablaufprotokoll für i und s. Was wird ausgegeben?

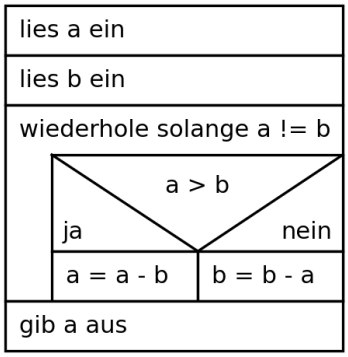


Lösung:

i	Start	1	2	3	4	5
s	0	1	0	3	2	7

Ausgabe: 7

6. (4 Punkte) Gegeben ist der folgende Algorithmus für positive ganze Zahlen a und b.



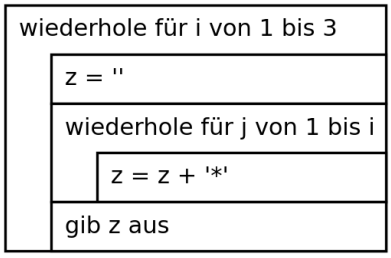
   (a) Erstelle ein Ablaufprotokoll für die Eingabe a = 12 und b = 18.
   (b) Was berechnet der Algorithmus?

Lösung:

Durchlauf	Start	1	2
a	12	12	6
b	18	6	6

Ausgabe: 6. Der Algorithmus berechnet den größten gemeinsamen Teiler von a und b.

7. (3 Punkte) Was gibt der folgende Algorithmus aus?



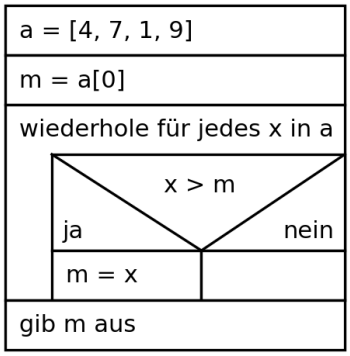
Lösung:

```
*
**
***
```

8. (3 Punkte) Zeichne ein Nassi-Shneiderman-Diagramm zu dem folgenden Programm.

```
a = [4, 7, 1, 9]
m = a[0]
for x in a:
    if x > m:
        m = x
print(m)
```

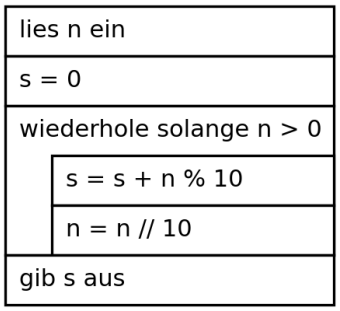
Lösung:



9. (3 Punkte) Zeichne ein Nassi-Shneiderman-Diagramm zu dem folgenden Pseudocode.

```
lies n ein
s = 0
solange n > 0:
    s = s + n % 10
    n = n // 10
gib s aus
```

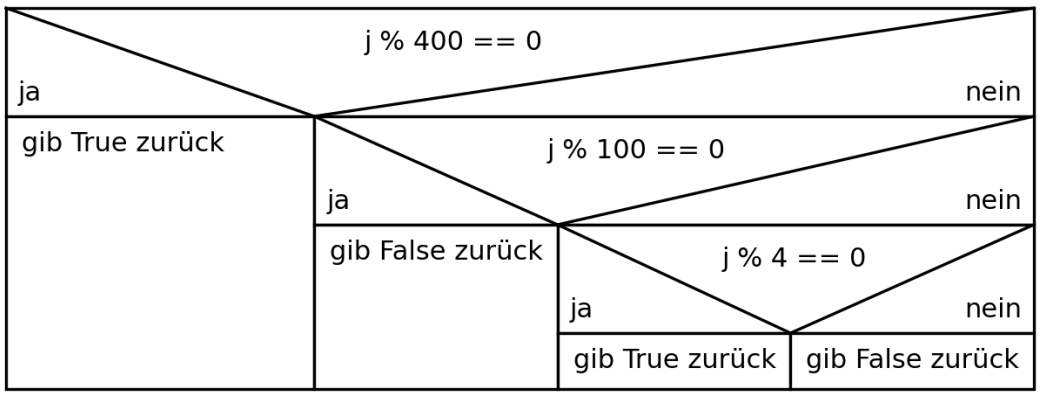
Lösung:



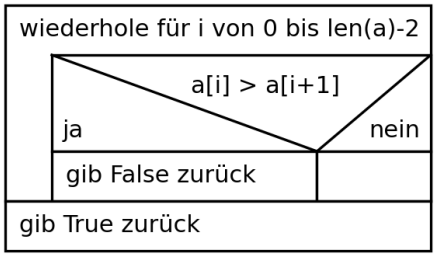
10. (4 Punkte) Zeichne ein Nassi-Shneiderman-Diagramm zu der folgenden Funktion.

```
def ist_schaltjahr(j):
    if j % 400 == 0:
        return True
    elif j % 100 == 0:
        return False
    elif j % 4 == 0:
        return True
    else:
        return False
```

Lösung:



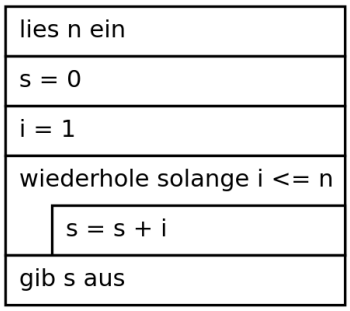
11. (3 Punkte) Der folgende Algorithmus gehört zu einer Funktion `ist_sortiert(a)`. Schreibe die Funktion in Python.



Lösung:

```
def ist_sortiert(a):
    for i in range(len(a)-1):
        if a[i] > a[i+1]:
            return False
    return True
```

12. (3 Punkte) Der folgende Algorithmus soll die Summe der Zahlen von 1 bis n ausgeben.



(a) Was passiert beim Ablauf für n = 3?

(b) Korrigiere den Algorithmus.

Lösung: a) Die Schleife endet nie, weil i nicht verändert wird (Endlosschleife).

b) Im Rumpf der Schleife fehlt unter `s = s + i` der Block `i = i + 1`.

13. (3 Punkte) Beantworte die folgenden Fragen.

(a) Nenne einen Vorteil eines Nassi-Shneiderman-Diagramms gegenüber Pseudocode.

(b) Warum beschreibt man einen Algorithmus oft mit Pseudocode und nicht gleich in Python?

(c) Was ist der Unterschied zwischen einer Solange-Schleife und einer Wiederhole-bis-Schleife?

Lösung: a) Die Struktur und die Schachtelung sind auf einen Blick sichtbar.
b) Pseudocode ist unabhängig von einer Programmiersprache und lässt Einzelheiten der Syntax weg.
c) Die Solange-Schleife prüft die Bedingung vor jedem Durchlauf, die Wiederhole-bis-Schleife danach. Ihr Rumpf läuft also mindestens einmal.