# Aufgabe 1

Spezifikation: Suchen in einer Liste von Namen:

Gegeben ist eine Liste L von Namen N. Stellen Sie fest ob N in der Liste L vorkommt!

Der Name besteht aus Name und Vorname. Die Reinfolge ist Nachname**,** Vorname. Ein Name kommt nur einmal in der Liste L vor. Das „,“ trennt Vor- und Nachname.

Ein Name besteht aus Groß- und Kleinbuchstaben des deutschen Alphabets und „-“ sowie Leerzeichen für Doppelnamen, wenn dies nicht erfüllt ist, ist der Suchname falsch. Die Länge für Vor- und Nachname sind jeweils auf 32 Stellen limitiert.

Wenn der Name nicht in der Liste L gefunden wird, wird falsch zurückgegeben. Wenn der Name gefunden wird, wird wahr zurückgegeben.

Die Liste kann nachträglich erweitert werden, die maximal Anzahl für Einträge (Vor- und Nachname) sind 100.000.000 (mehr als die Population von Deutschland)

# Aufgabe 2

## a)

Die Multiplikation kann durch die Addition dargestellt werden, da Multiplikation eine abkürzende Schreibweise für die Addition von gleichen Summanden ist. 2+2+2+2+2+2+2 ist das Gleiche wie 2\*7, die 7 gibt an, dass die Zahl 2 7mal addiert wird. Im Natürlichem Zahlenraum ist dies durch die rekursive Definition von Peano und dessen Axiome bewiesen. Eine Auswahl dieser besagen:

1) ∀n (n ∈ N →n' ∈ N) Jeder Zahl hat einen Nachfolger (‘ heißt Nachfolger)

2) 0 ∈ N Das Neutrale Element 0 gehört zu den Natürlichen Zahlen

3) 1:= 0‘ Die Zahl 1 ist der Nachfolger der 0

Additionsdefinition: n‘ = n+1

Rekursive Definition der Multiplikation:

n \* m‘ := (n\*m) + n

Demnach das Beispiel: 2\*4

n = 2, m = 4

2\*4 = (2\*3)+2 = ((2\*2) +2 )+2 = (((2\*1) +2 )+2)+2 = ((((2\*0) +2 ) +2)+2)+2 = 2+2+2+2 = 8

## b) Algorithmus

### a) Schleife

Erster Gedanke:

multiplikationDurchAddition**(**↓Integer n**,** ↓Integer m**,** ↑Integer result**)**

result = 0

**while** **(**i **<=** m**)**

result **=** result **+** n

i = i + 1

end **while**

Ausgabe result

end multiplikationDurchAddition

Verbesserung:

multiplikationDurchAddition**(**↓Integer n**,** ↓Integer m**,** ↑Integer result**)**

result = 0

**while** **(**m > 0**)**

result **=** result **+** n

m = m - 1

end **while**

Ausgabe result

### b) Rekursiv nach Definition von Peano

Geht über das im Unterricht gemachte heraus. Ich habe es hier jedoch angefügt, da dies den Gedanken aus Aufgabenteil a) darstellt.

multiplikationDurchAdditionRekursiv **(**↓Integer n**,** ↓Integer m**,** ↑Integer result**)**

**if** **(**m **>** 0**)** then

Ausgabe result **=** multiplikationDurchAdditionRekursiv**(**↓n**,** ↓m**-**1**,** ↑result**)** **+** n

// Das Ergebnis (result) ist das Ergebnis des gleichen Algorithmus

// mit dem zweiten Multiplikand um 1 reduziert und dazu n addiert.

end **if**

**if** **(**m **==** 0**)**

Ausgabe result **=** 0

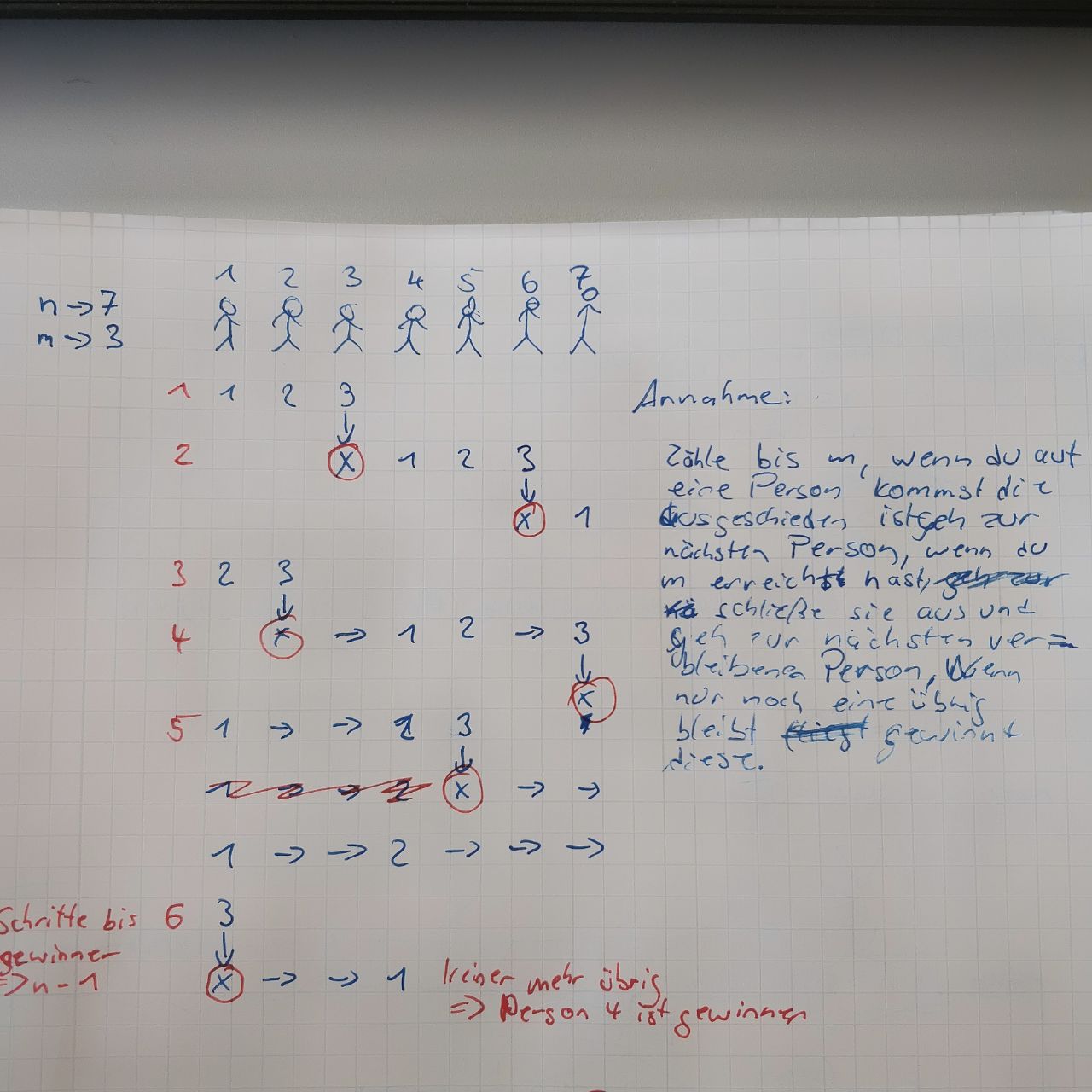
end **if**

end multiplikationDurchAdditionRekursiv

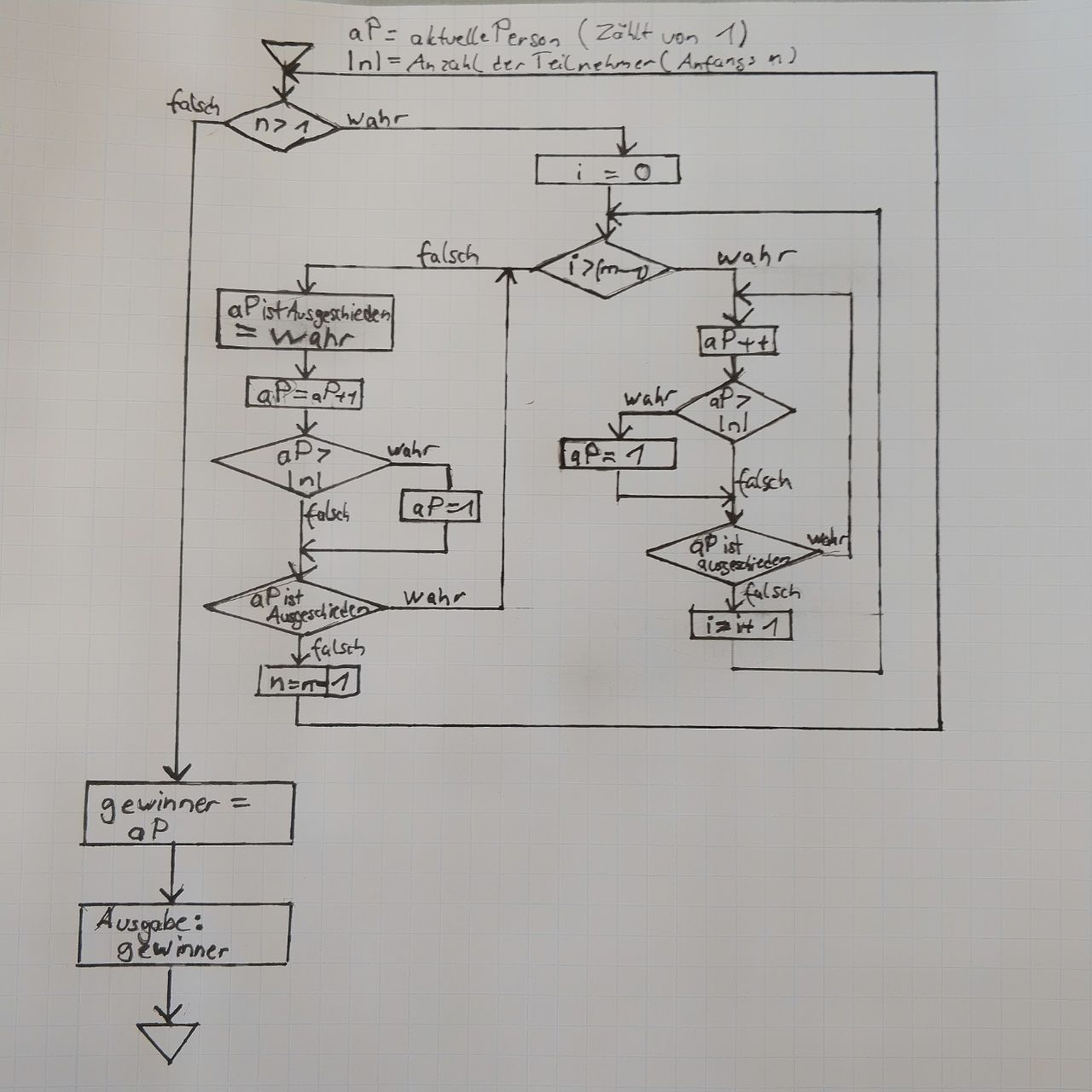
# Aufgabe 3

Herangehensweise:

Vorstellung im Kreis -> Personen in einer Reihe. Bei n=7 ist Person 8 die gleiche wie 1.



Endgültiger Algorithmus:



Test mit Zahlen:

n = 4, m = 2, |n|=4 Erwartung: Gewinner ist Person 1

01 aP = 1

02 n > 1 -> wahr

03 i = 0

04 i > (m-1) -> wahr

05 aP++ -> aP = 2

06 aP>|n| -> falsch

07 aPistausgeschieden -> falsch

08 i = i+1 -> i=1

08 i > (m-1) -> falsch

09 aPistausgeschieden = true //Spieler 2 ausgeschieden

10 aP = aP+1 -> aP = 3

11 aP>|n| -> falsch

12 aPistausgeschieden -> falsch

13 n = n-1 -> n=3

14 n>0 -> wahr

15 i = 0

16 i > (m-1) -> wahr

17 aP++ -> aP = 4

18 aP>|n| -> falsch

19 aPistausgeschieden -> falsch

20 i = i+1 -> i=1

21 i > (m-1) -> falsch

22 aPistausgeschieden = true //Spieler 4 ausgeschieden

23 aP = aP+1 -> aP = 5

24 aP>|n| -> wahr

25 aP=1

26 aPistausgeschieden -> falsch

27 n = n-1 -> n=2

28 n>0 -> wahr

29 i = 0

30 i > (m-1) -> wahr

31 aP++ -> aP = 2

32 aP>|n| -> falsch

33 aPistausgeschieden -> wahr

34 aP++ -> aP = 3

32 aP>|n| -> falsch

33 aPistausgeschieden -> falsch

34 i = i+1 -> i = 1

35 i > (m-1) -> falsch

36 aPistausgeschieden = true //Spieler 3 ausgeschieden

37 aP = aP+1 -> aP = 4

38 aP>|n| -> falsch

39 aPistausgeschieden -> wahr

40 aPistausgeschieden = true

41 aP = aP+1 -> aP = 5

42 aP>|n| -> wahr

43 aP=1

44 aP>|n| -> falsch

45 aPistausgeschieden -> falsch

46 n = n-1 -> n=1

47 n > 1 -> falsch

48 gewinner = aP -> gewinner = 1

49 Ausgabe gewinner