1 merge

```
merge :: ([Integer], [Integer]) -> [Integer]
merge ([], b) = b
merge (a, []) = a
merge (a:a1, b:b1) = if a < b then
                         a : merge (a1, b:b1)
                    else b : merge (a:a1, b1)
```

Die Funktion ist definiert auf einem Tupel aus Integer-Listen, das heißt, wir wissen auf jeden Fall

$$P \subseteq [\text{Integer}] \times [\text{Integer}]$$

eine größere Parametermenge verbietet uns das Typsystem.

Nach Aufgabenstellung können wir uns auf endliche Listen beschränken, d.h

$$P \subseteq [\text{Integer}] \times [\text{Integer}], \text{ mit } [\text{Integer}] \text{ endlich}$$

Oder auch:

$$P \subseteq \{(a,b)|a,b \in [\text{Integer}] \wedge a,b \text{ endlich}\}$$

Wenn wir uns die rekursiven Aufrufe ansehen, erkennen wir, dass immer ein e Liste der beiden verkürzt wird, daher bietet sich als Deltafunktion die gesammte Länge, d.h. die Summe der beiden Längen an. Die Noethersche Ordnung wäre damit $(\mathbb{N}, \leq)$ Außerdem sehen wir, dass wir den Parameterbereich nicht weiter einschränken brauchen, somit haben wir jetzt:

$$N = (\mathbb{N}, le)$$
$$P = \{(a,b)|a,b \in [\text{Integer}] \wedge a,b \text{ endlich}\}$$

$$\delta : P \to N$$
$$\delta(a,b) = \text{length}\, a + \text{length}\, b$$

Um mit dieser δ-Funktion jetzt etwas untersuchen zu können, sollten wir aber genauer aufschreiben, wie die Funktionen G_1 und G_2, die die Veränderung des formalen Parameters für den rekursiven Aufruf beschreibt, aussieht:

$$G_1 : P \to P$$
$$G_1(x,y) = (\text{tail}\, x, y)$$

$$G_2 : P \to P$$
$$G_2(x,y) = (x, \text{tail}\, y)$$

Jetzt müssen wir *erstens* Prüfen, dass die rekursiven aufrufe wieder auf zulässigen Parametern erfolgen:

Da die Listen endlich lange sind und im Falle eines rekursiven Aufrufs mindestens ein Element haben, und der rekursiven Aufruf jeweils nur um ein tail verändert ist, bleiben wir in P.