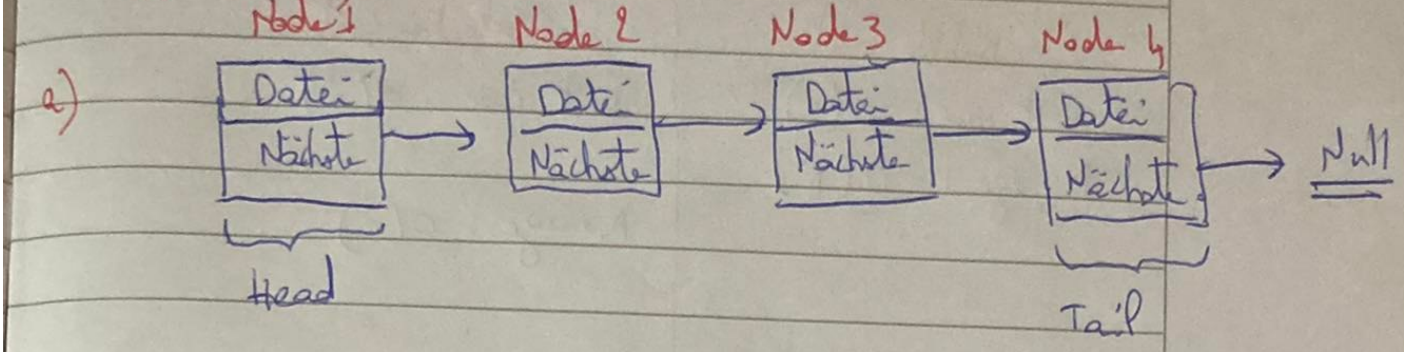
# Algorithmen und Datenstruktur:

## Aufgabe 1:



b) - Vor- und Nachnamen sind natürlich ~~als~~ vom Datentyp String.
Da "Tom" als vorname oder "Hanks" als Nachname kann weder als Integer dargestellt werden, noch als Char.
Nur in dem Fall, wo wir es als Bytes repräsentieren wollen, was hier ganz kompliziert und nicht „user-friedly" ist..

- Kurs und Matrikelnummer habe ich als Integers dargestellt, da eine MatrikelNUMMER ein Nummer ist, und die Kurse sind generell, also man kann eine Zahl für jede Kurs vordefinieren, und es macht den ganzen Dinge ganz effizienter zu codieren.

## Aufgabe 2 (Code)

## Aufgabe 3:

- Die Methoden, die bei der Verketteten Liste effizient sind:

* add(T data, int index), da mithilfe von Knoten wird es ganz einfacher, die Ordnung ~~B~~ der Liste zu verändern. Man wählt das Index.

- Die Methoden, die ineffizienter sind:

* add(T data): In einen Array kann man mit ~~pt~~ push(Element) direkt das neue Element am Ende hinzufügen, aber in einer Verketteten liste muss man sicherstellen, dass die Indexe korrekt sind.

~~B~~ * RemoveAll: man muss durch die ganze Liste gehen, Schleife nach Schleife. In Array setzt man length = 0.

Aufgabe 4: (Code)

Aufgabe 5:

Komplexität

Bubble Sort

Best case: $O(n)$

Average: $O(n^2)$

Selection Sort

Best case: $O(n^2)$

Average: $O(n^2)$