Call by value vs. Call by reference

```
void init(List *list) {
    list = (List *)malloc(sizeof(List));
    assert(list);
    list->first = NULL;
    list->last = NULL;
}
```

Call by value! Funktioniert nicht wie gewünscht, da der Zeiger `list` **kopiert** wird.

Mögliche Implementationen:

```
void init(List **list) {
    assert(list);
    (*list) = (List *)malloc(sizeof(List));
    assert(*list);
    (*list)->first = NULL;
    (*list)->last = NULL;
}
```

oder:

```
List* init() {
    List *list;
    list = (List *)malloc(sizeof(List));
    assert(list);
    list->first = NULL;
    list->last = NULL;
    return list;
}
```

Conditional AND

```
If ((item != NULL) && (item->val == 15)) {
    ...
}
```

Bei einem **Conditional AND** wird der zweite Ausdruck nur ausgewertet, wenn der erste **TRUE** war.

Typischer C-Fehler

```
if (list->last = NULL) {
    list->last = list->first;
}
```

x==y liefert einen Wahrheitswert. Da in C jeder Wert ein Wahrheitswert sein kann, liefert aber auch x=y einen Wahrheitswert. Solche Fehler erkennt der Compiler nicht!