

Aufgaben

1) Zeichenkette umkehren

Schreibe ein Programm, das eine gegebene Zeichenkette (String) umkehrt. Leerzeichen und Sonderzeichen sollen unverändert als Zeichen behandelt werden.

Anforderungen:

- Eingabe: eine Zeichenkette *s*.
- Ausgabe: die umgekehrte Zeichenkette von *s*.
- Handle auch Randfälle (leere Zeichenkette, 1 Zeichen).

Beispiele:

Eingabe: "Hallo Welt!"
Ausgabe: "!tleW ollaH"

Eingabe: ""
Ausgabe: ""

2) Worthäufigkeiten zählen (Wort → Anzahl)

Schreibe ein Programm, das für einen gegebenen Text die Häufigkeit jedes Wortes ermittelt und als Zuordnung *Wort* → *Anzahl* ausgibt.

Vorgaben:

- Groß-/Kleinschreibung ignorieren (alle Wörter in Kleinbuchstaben zählen).
- Satzzeichen nicht mitzählen (z. B. „Test,“ wird als „test“ gezählt).
- Umlaute/ß dürfen als normale Zeichen behandelt werden.

Beispiel:

Eingabe-Text:
"Das ist ist ein Test. Das ist nur ein Test!"

Erwartete Ausgabe (Reihenfolge beliebig):
das: 2
ist: 3
ein: 2
test: 2
nur: 1

3) Infix-Prüfung zwischen zwei Zeichenketten

Schreibe ein Programm, das für zwei Zeichenketten *s1* und *s2* prüft, ob eine der beiden ein *Infix* (Teilstring) der anderen ist. Gib *true* zurück, wenn *s1* in *s2* vorkommt oder *s2* in *s1* vorkommt; sonst *false*.

Beispiele:

s1 = "abc", *s2* = "zabcy" → true ("abc" ist Infix von *s2*)
s1 = "haus", *s2* = "maus" → true ("aus" ist Infix in beiden Strings)
s1 = "xyz", *s2* = "abcy" → false

4) Alle möglichen Wortaufteilungen ohne Leerzeichen

Gegeben ist eine Zeichenkette `s` ohne Leerzeichen sowie ein Wörterbuch `D` (Menge gültiger Wörter).
Schreibe ein Programm, das **alle** möglichen Aufteilungen von `s` in eine Folge von Wörtern aus `D` ausgibt.

Hinweise:

- Wenn keine Aufteilung möglich ist, gib eine leere Liste aus.
- Es können sehr viele Aufteilungen existieren – verwende daher eine effiziente Strategie (z. B. Rekursion mit Memoisierung oder dynamische Programmierung).

Beispiel:

```
s = "diesisteintest"
D = {"dies", "die", "s", "ist", "ein", "in", "test"}
```

Mögliche Ausgaben (Reihenfolge beliebig):

```
["dies", "ist", "ein", "test"]
["die", "s", "ist", "ein", "test"]
```

Tipp: Für die Aufgaben 2–4 lohnt sich das Schreiben von Unit-Tests mit aussagekräftigen Randfällen.