# Hauptstädte-Galgenmännchen

In diesem Projekt geht es darum ein kleines Spiel zu entwickeln in dem der Spieler die Hauptstadt zu einem angegebenen Land erraten muss.

Zunächst wird dem Spieler das Land und eine maskierte Version der Hauptstadt angezeigt. Die maskierte Version zeigt für jeden Buchstaben des Namens der Hauptstadt einen Unterstrich an.

Wenn der Spieler danach eine Taste drückt wird entweder der Buchstabe an der entsprechenden Stell angezeigt oder der Fehlerzähler um eins erhöht, bis alle Buchstaben sichtbar sind.

### Beispiel

```
Italien
_ _ _ (0 Fehler)
```

1. Spieler drückt `R`

```
Italien
_ _ _ (0 Fehler)
R _ _ (0 Fehler)
```

2. Spieler drückt `I`

```
Italien
_ _ _ (0 Fehler)
R _ _ (0 Fehler)
R _ _ (1 Fehler)
```

## Aufgabe 1: Dateien einlesen

Implementieren Sie die Funktionen in `capitalsdata.c`. Beachten Sie die nachfolgenden Hinweise.

### Dateiformat

Die erste Zeile enthält die Anzahl der Land-Hauptstadt-Paare. Die nachfolgenden Zeilen enthalten abwechselnd ein Land und die zugehörige Hauptstadt.

**Beispiel:**

```
3
land1
hauptstadt1
land2
hauptstadt2
land3
hauptstadt3
```

**Hinweise**

- Die Dateien enthalten keine Leerzeilen und keine Leerzeichen die nicht in dem Namen einer Stadt oder eines Landes enthalten sind.
- Im Ordner `data` gibt es zwei Dateien, jeweils Länder und die zugehörigen Hauptstädte auf Deutsch und Englisch enthalten.
- Machen Sie sich mit den Funktionen `fopen`, `fclose` und `fscanf` vertraut.
- Sie dürfen die bereits implementierte Funktion `read_line` benutzen.

*Der Test `test_data_io` deckt diese Aufgabe ab.*

## Aufgabe 2: Spielmechanik implementieren

Implementieren Sie die Funktionen in `game.c` (Außer `printGameState`).

**Hinweise**

- Es sollen nur alphabetische Zeichen (a-z und A-Z) maskiert werden, alle anderen Zeichen sind von Anfang an sichtbar und müssen nicht erraten werden.
- Zwischen Groß- und Kleinschreibung soll beim Aufdecken eines Zeichens nicht unterschieden werden.
- Das Spiel ist beendet sobald alle Zeichen aufgedeckt wurden.

*Der Test `test_game` deckt diese Aufgabe ab.*

## Aufgabe 3: Spiel fertigstellen

Setzen Sie die Komponenten zum fertigen Spiel zusammen:

1. Implementieren Sie `printGameState` so, dass es das maskierte Lösungswort ähnlich dem Beispiel oben in der Konsole anzeigt.

2. Implementieren Sie `main`. Beim Start des Programms soll eine Eingabedatei eingelesen werden und zufällig ein Land ausgewählt werden. Danach wartet das Programm auf die Eingabe eines Zeichens. Wenn ein Zeichen eingegeben wurde wird es entweder aufgedeckt oder der Fehlerzähler erhöht. Dabei wird der Zustand des Spiels ähnlich dem Beispiel oben in der Konsole angezeigt.

**Hinweise**

- Machen sie sich mit den Funktionen `printf`, `rand` und `getchar` vertraut.

## Bonusaufgaben

- Setzen Sie ein Limit für Fehler. Schreiben Sie eine Rangliste der Spieler mit den meisten erratenen Wörtern in eine Datei.
- Geben Sie dem Spieler Tipps, wenn er wiederholt Fehler macht.
- Versuchen Sie Ihren Code mit verschiedenen Sprachen kompatibel zu machen.