Prof. Dr. Franz Korf
HAW Hamburg

# C-Eingangstest für die Veranstaltung Betriebssysteme in der Technischen Informatik

**Vorbemerkung** Die Aufgaben des Praktikums zur Vorlesung Betriebssysteme basieren auf C. Folgende Aufgabe umfasst das Minimum an C Kenntnissen, die für die Teilnahme am Praktikum wichtig sind. Überprüfen Sie anhand dieser Aufgabe Ihre C Kenntnisse und holen Sie ggf. fehlendes Wissen noch vor dem ersten Praktikumstermin nach.

Eine mögliche **Lösung** dieser Aufgabe liegt bei. Bitte programmieren Sie die Aufgabe selbst und vergleichen Sie Ihr Programm mit der beiliegenden Lösung.

Die **Aufgabe** ist an das **8 Damen Problem** angelehnt. Bei diesem Problem müssen acht Damen so auf einem Schachbrett platziert werden, dass keine zwei Damen sich gegenseitig bedrohen – schlagen können. Zwei Damen bedrohen einander, wenn sie

- in derselben Zeile stehen oder
- in derselben Spalte stehen oder
- auf derselben Diagonale stehen.

**Beispiel**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| X | | | | | | | |
| | | | | | | X | |
| | | | | X | | | |
| | | | | | | | X |
| | X | | | | | | |
| | | | X | | | | |
| | | | | | X | | |
| | | X | | | | | |

Für das 8 Damen Problem gibt es 92 Lösungen (inklusive symmetrische Lösungen).

In dieser Aufgabe wird das Problem für n Damen gelöst, d.h.: **Auf einem n x n Schachbrett sollen n Damen so platziert werden, dass sie sich nicht schlagen können.** n wird dem Programm als **optionaler Parameter** übergeben. Der default Wert ist 8.

Sie können gerne Ihr eigenes Modulkonzept entwickeln und umsetzen. Alternativ können Sie sich an nachfolgende Rahmenbedingungen halten. Wichtig ist, dass folgende C Elemente verwendet werden:

- Zeiger und Felder
- Mehrere Module
- Ein- und Ausgabe von Text
- C structs
- Speicher vom Heap anfordern und wieder frei geben
- Fehlerbehandlung
- Standard-Kontrollstrukturen

**Hinweise für eine mögliche Umsetzung der Aufgabe**

- Erstellen Sie ein Modul **board** (`board.h und board.c`), das die Grundfunktionen rund um das Schachbrett enthält. Dazu zählen
  - Erstellen und Löschen eines Schachbretts auf dem Heap
  - Ausgabe eines Schachbretts (z.B. als ASCII Grafik)
  - Setzen von Figuren auf dem Schachbrett
  - Abfragen der Belegung eines Felds auf dem Schachbrett

- Das Schachfeld könnte durch folgende C Struktur modelliert werden

```
typedef struct {
      int size;         // Dimension des Felds
      char *board;      // z.B. 2-dimensionales VLA
} BOARD, *P_BOARD;
```

  wobei size die Größe des Feldes und das brett das Spielfeld selbst abspeichert.

- Erstellen Sie ein Modul **checkers** (`checkers.h und checkers.c`), welches das n-Damen Problem löst. Es greift auf das Modul `board` zu. Insbesondere sollte das Modul folgende Funktionen enthalten:
  - Eine Funktion die überprüft, ob die Positionierung einer Dame auf einem bestimmten Feld zur Bedrohung schon platzierter Damen führt.
  - Eine Funktion, die alle Lösungen des n-Damen Problems mit **Backtracking** berechnet. Sie liefert die Anzahl der Lösungen zurück. Mit Hilfe der Ausgabefunktion des Moduls `board` gibt sie optional die Lösungen auf `stdout` aus.

- **Konzept des Backtracking Algorithmus**: Der Lösungsraum wird systematisch durchsucht. Dabei wird eine Lösung schrittweise aufgebaut. Beim n-Damen Problem bedeutet dies:
  - i Damen sind auf dem Schachbrett platziert und bedrohen sich nicht. Im nächsten Schritt wird eine weitere Dame hinzugefügt und überprüft, ob diese die anderen Damen auf dem Spielfeld bedroht.
  - Ein Schritt wird rückgängig gemacht, sobald erkannt wird, dass die zugehörige Teillösung nicht zu einer Lösung vervollständigt werden kann. Somit werden Lösungskandidaten, die aus dieser Teillösung hervorgehen würden, nicht weiter betrachtet.

- **Anwendung von Backtracking auf das n Damen Problem:** Es ist offensichtlich, dass in jeder Zeile maximal eine Dame stehen darf (zwei Damen in einer Zeile bedrohen sich gegenseitig). Damit n Damen auf dem n x n Schachbrett platziert werden können, muss in jeder Zeile genau eine Dame stehen (ansonsten stehen zwei oder mehr Damen in einer Zeile und bedrohen sich gegenseitig).
  Der Backtracking Algorithmus geht dann wie folgt vor:
  - In den ersten i Zeilen ist je eine Dame platziert. Die Damen bedrohen sich nicht.
  - Im nächsten Schritt wird eine Dame in der Zeile i+1 so platziert, dass **diese** keine anderen Damen auf dem Spielfeld bedroht.
  - Kann keine Dame in Zeile i+1 entsprechend platziert werden, führt diese Teillösung nicht zum Erfolg. Sie wird nicht weiter betrachtet. Nun wird die Dame in Zeile i auf der nächsten möglichen Position in dieser Zeile platziert.
    Dieser Schritt beinhaltet die Backtracking Komponente: Schritte werden rückgängig gemacht, sobald man erkennt, dass die zugehörige Teillösung in eine Sackgasse führt. Sind in den

ersten i+1 Zeilen die Damen so platziert, dass sie sich gegenseitig bedrohen, muss diese Teillösung nicht weiterverfolgt werden.
    - Somit bietet sich für die Lösung eine rekursive Funktion an.

- Damit müssen in der Funktion **`main`** nur noch die Komponenten passend zusammengefügt werden. Der optionale Eingabeparameter – Größe des Schachfelds – wird überprüft, die entsprechende Funktion des Moduls `checkers` wird aufgerufen und das Ergebnis wird ausgegeben.

**Bitte beachten Sie bei der Erstellung der Lösung** typische Punkte, die Sie bisher gelernt haben. Dazu zählen zum Beispiel das Einhalten eines Codeing Styles, sinnvoll gewählte Namen, Refactoring, Regeln zum Aufbau der Schnittstelle eines Moduls oder Anforderungen der Art „eine Funktion erfüllt nur eine Aufgabe“.

Zur Orientierung gibt folgende Tabelle für unterschiedliche Dimensionen die Anzahl der Lösungen an.

| Größe des Schachfelds | Anzahl der Lösungen |
|---|---|
| 1x1 | 1 |
| 2x2 | 0 |
| 3x3 | 0 |
| 4x4 | 2 |
| 5x5 | 10 |
| 6x6 | 4 |
| 7x7 | 40 |
| 8x8 | 92 |
| 9x9 | 352 |
| 10x10 | 724 |

Bitte melden Sie sich via Mail oder Slack (haw-hh-ti-gt-gs-bs.slack.com) bei Problemen und Fragen, einem potenziellen Fehler in der Tabelle oder Problemen in der beiliegenden Lösung.

**Viel Spaß bei diesem kleinen C Warming-Up.**