Prof. Dr. Visvanathan Ramesh
PD. Dr. Arne Nägel
Lukas Müller

Goethe-Universität Frankfurt am Main
Institut für Informatik

Programmierparadigmen und Compilerbau

Sommersemester 2022

Aufgabenblatt Nr. 1

Abgabe Mittwoch 04.05.2022, 10 Uhr in Moodle!

Aufgabe 0 **(0 Punkte)**

Lesen Sie sich die folgenden Hinweise gut durch!

- Gruppenabgaben sind **nicht erlaubt!** Die Aufgaben dürfen aber gerne in Gruppen gelöst werden, lediglich das aufschreiben der Aufgaben soll **einzeln** erfolgen. Plagiate werden nicht toleriert.
- Bitte schreiben Sie in jede Datei die Sie abgeben wollen den **Name** und die **Matrikelnummer**.
- Zu implementierende Programme sind stets auch zu dokumentieren und zu testen. Testaufrufe mit geeigneten Werten sind der Lösung als Kommentare beizufügen!
- Fassen sie alle Dateien die Sie Abgeben wollen in eine einzige .zip Datei auf Moodle hoch. Achten Sie Bitte darauf, das sie nur .pdf und .hs Dateien abgeben.
- Der Dateiname soll dem Muster `Name_Gruppe_Gruppennummer_Blatt_Blattnummer.Dateiendung` entsprechen.

Aufgabe 1 **(3+3+3 = 9 Punkte)**

Die Funktion `f` sei definiert als

```
f a b c = if b >= 6
        then (if b < 23 then a*3 else f (a+2) (b+3) c)
        else f (a+2) (b+3) c
```

Geben Sie für alle Teilaufgaben jeweils sämtliche Reduktionsschritte sowie die jeweils verwendete Regel als Buchstabe **D, A** oder **I** an (**D**=Definitionseinsetzung, **A**=Arithmetische Auswertung, **I**=if-Auswertung).

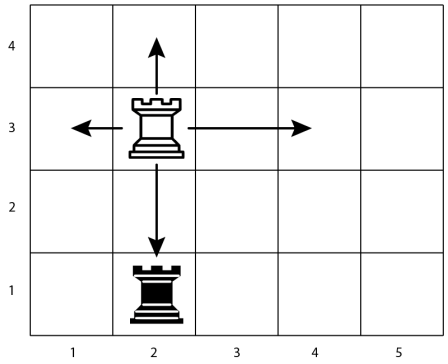
a) Werten Sie `f 1 2 (f 12 42 26)` in *normaler Reihenfolge* aus.

b) Geben Sie die ersten 14 Schritte der Auswertung von `f 1 2 (f 12 42 26)` in *applikativer Reihenfolge* an. Wie setzt sich die Berechnung danach weiter fort? Begründen Sie Ihre Antwort in einem Satz, die Fortsetzung der Rechnung ist dafür nicht erforderlich.

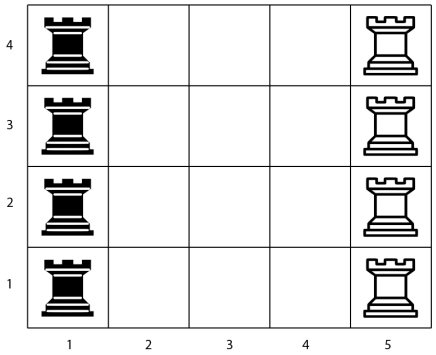
c) Werten Sie `f 10 3 (f 15 30 1337)` in *verzögerter Reihenfolge* aus.

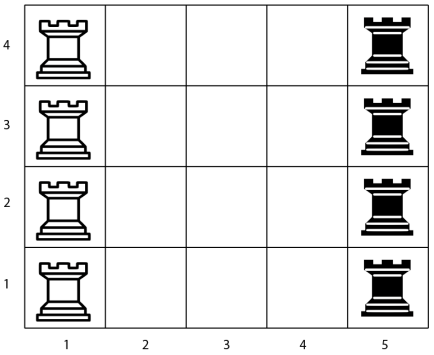
Aufgabe 2 **(2+2+2+3+2 = 11 Punkte)**

In dieser Aufgabe betrachten wir ein Rätsel auf einem 5×4 großen Schachbrett. Zur einfacheren Implementierung sind die Felder nicht wie sonst üblich durch eine Kombination aus Buchstabe und Ziffer benannt, sondern mit zwei Ziffern, wobei zuerst die horizontale Position genannt wird und dann die vertikale – statt d3 verwenden wir also 43 als Feldangabe. Auf dem Schachfeld befinden sich nur Türme.
Der Turm darf sich nur horizontal oder vertikal bewegen und darf über andere Türme springen. Auf einem Feld darf sich zu jedem Zeitpunk maximal ein Turm befinden und es dürfen keine Türme der gegnerischen Farbe geschlagen werden. Falls ein Turm einen gegnerischen Turm schlagen könnte, kann auch gesagt werden, das diese den anderen Turm *bedroht*.



Das Ziel des Rätsels ist das Finden einer Zugfolge, um ausgehend von der Stellung in der linken Abbildung die Stellung in der rechten Abbildung zu erreichen, wobei in jedem Zug ein beliebiger Turm ziehen darf (die Farbe spielt keine Rolle). Dabei sollen **nur** die Teilaufgaben bearbeitet werden.





Das Spielfeld entspricht einer Matrix, die durch eine zweistellige Funktion f (z.B. `feldA`) modelliert wird, wobei $f\ x\ y$ das Matrix-Element in Spalte x und Zeile y darstellt. Dabei steht ein Matrix-Element mit Wert `'w'` für einen weißen Turm, `'s'` für einen schwarzen Turm und `' '` für ein leeres Feld. Beachten Sie, dass man im GHCi keine Funktion anzeigen kann. Zur Anzeige eines Spielfelds kann die Funktion `zeigeFeld` verwendet werden, die in der Datei `blatt1.hs` definiert ist und ein Spielfeld auf die Kommandozeile ausgibt. Diese Datei findet sich beim Aufgabenblatt 1 auf der Webseite zur Vorlesung. Die in dieser Datei definierte Funktion `feldA` entspricht dem linken Spielfeld. `feldB` entspricht einer Stellung vor dem rechten Abbildung.

a) Implementieren und testen Sie in Haskell eine Funktion `istZugValide`, die als erstes Argument die x-Koordinate der zu ziehenden Figur, als zweites die dazugehörige y-Koordinate, als drittes die x-Koordinate des Zielfelds und als viertes die dazugehörige y-Koordinate enthält und genau dann `True` zurückliefert, falls der Zug horizontal oder vertikal ist.

Beispielaufruf:

```
*Main> istZugValide 1 1 1 3
True
*Main> istZugValide 1 1 2 2
False
```

b) Implementieren und testen Sie in Haskell eine Funktion `bedroht`, die als erstes Argument eine x-Koordinate, als zweites eine y-Koordinate, als drittes die Farbe der Figur (`'w'`oder `'s'`) und als viertes das Spielfeld erhält und `True` genau dann zurückgibt, wenn auf dem Feld eine Figur des anderen Spieler ist, und somit die Figur schlagen könnte.

Beispielaufruf:

```
*Main> bedroht 5 1 's' feldA
True
*Main> bedroht 4 1 's' feldA
False
```

c) Implementieren und testen Sie in Haskell eine Funktion `gueltigerZug`, die als erstes Argument die x-Koordinate der zu ziehenden Figur, als zweites die dazugehörige y-Koordinate, als drittes die x-Koordinate des Zielfelds, als viertes die dazugehörige y-Koordinate und als fünftes das Spielfeld erhält und genau dann `True` zurückliefert, falls der Zug gemäß allen Regeln gültig ist.
Hinweis: Dabei soll auch berücksichtigt werden, ob die Koordinaten passen.

Beispielaufruf:

```
*Main> gueltigerZug 1 1 3 1 feldA
True
*Main> gueltigerZug 1 1 1 3 feldA
False
```

d) Implementieren und testen Sie in Haskell eine Funktion `zieheWennGueltig`, die als erstes Argument die x-Koordinate der zu ziehenden Figur, als zweites die dazugehörige y-Koordinate, als drittes die x-Koordinate des Zielfelds, als viertes die dazugehörige y-Koordinate und als fünftes das Spielfeld erhält und das Spielfeld nach Ausführung des Zuges zurückliefert. Falls der Zug nicht gültig ist, soll das Spielfeld unverändert zurückgegeben werden.
Tipp: Das Ergebnis von `zieheWennGueltig` ist eine Funktion mit zwei Argumenten. Hier bietet sich eine anonyme Funktion, also eine Funktion ohne Namen an: `(\a b-> ...)` Unter welchen Bedingungen sind die Werte des gegebenen Spielfelds zu übernehmen?

Beispielaufruf:

```
*Main> zeigeFeld (zieheWennGueltig 1 1 3 1 feldA)
s   w
s   w
```

```
s   w
  s w
*Main> (zieheWennGueltig 1 1 3 1 feldA) 3 1
's'
```

e) Implementieren und testen Sie in Haskell eine Funktion gelöst, die für ein Spielfeld `True` genau dann zurückliefert, wenn das Rätsel gelöst ist. FeldB ist in der baltt_1.hs definiert.

Beispielaufruf:

```
*Main> geloest (zieheWennGueltig 4 2 5 2 feldB)
True
*Main> geloest (zieheWennGueltig 4 2 3 2 feldB)
False
```