



Technische Universität Berlin

Software and Embedded Systems Engineering Group

Prof. Dr. Sabine Glesner

www.sese.tu-berlin.de Sekr. TEL 12-4 Ernst-Reuter-Platz 7 10587 Berlin






Softwaretechnik und Programmierparadigmen WiSe 2023/2024

Prof. Dr. Sabine Glesner
Simon Schwan
Julian Klein

Übungsblatt 02

Aufgabe 1: Rekursion II



Letzte Woche habt ihr die `win` Funktion implementiert. Jetzt soll diese Funktion etwas verbessert werden.

- a) De Casinobetreiber*innen ist der Code nicht hübsch genug, was durch Pattern Matching in der Funktion `win` verbessert werden könnte. Erweitert die bisherige Implementierung entsprechend. 
- b) Entwerft und implementiert eine optimierte Version der `win` Funktion, damit sie weniger Speicher verbraucht. 
- *) Zusatz: Um Besucher*innen möglichst lange am Spielen zu halten, sollen am Automaten Einsätze vorgeschlagen werden, die vorangegangene Verluste scheinbar größtenteils wieder ausgleichen können. Die Fibonacci-Folge hat sich dafür als wirkungsvoll erwiesen. Implementiert eine Funktion `fib :: Int -> Int` und überlegt euch eine mögliche Optimierung. 





Aufgabe 2: Eigene Datentypen

Euer Casino-Rechner beherrscht nun einige Grundfunktionen. Überlegt euch einen geeigneten Datentypen, um Anweisungen zu repräsentieren. Verwendet `deriving Show`, damit





Schlüssel:

-  Ein ergänzendes Video wird zur Vor- oder Nachbereitung veröffentlicht.
-  Wird im Tutorium besprochen.

Werte der Typen auf der Konsole so ausgegeben werden können wie sie eingegeben werden.

- a) Zunächst soll mit einem Aufzählungstyp `CommandS` nach der gewünschten Berechnung zwischen `Put`, `Take` und `Win` unterschieden werden. Die Berechnungen sollen mit einer Funktion `eval :: CommandS -> Int -> Int -> Int` durchgeführt werden können. 
- b) In einer ersten Erweiterung sollen die einzelnen Anweisungen als Summen- und Produkttyp `CommandP` über ihre Parameter verfügen. Erweitert auch die `eval` Funktion entsprechend. Wie verändert sich der Typ der Funktion `eval`? 
- c) Nun sollen auch noch beliebig komplexe Berechnungen unter Verwendung der drei Funktionen in einem rekursiven Datentyp `CommandR` dargestellt werden, in dem die Operanden jeder Operation entweder wieder Operationen oder Werte `ValR` sein können. Die neue Struktur benötigt eine rekursive `eval` Funktion, die den gesamten Baum auswerten kann.  

Aufgabe 3: Arbeiten mit Strings

- a) Haskell verwendet Unicode. Gebt alle ASCII-Zeichen (die ersten 7 Bit) mithilfe einer Range aus¹. 
- b) Einzelne Zeichen können auch im Pattern Matching verwendet werden. Schreibt eine Funktion `isVowel :: Char -> Bool`, die überprüft, ob ein gegebenes Zeichen ein (kleingeschriebener) Vokal ist. 
- c) Erweitert die Funktion zu `hasVowel`, die überprüft, ob ein gegebener String einen Vokal enthält. 
- d) Schreibt eine Funktion `toString`, mit der ein `CommandR` in einen String umgewandelt wird, so dass es kompakter auf der Konsole ausgegeben werden kann. Dabei sollen die Rechenoperationen möglichst “normal” aussehen. 

¹ `'A' = '\65'`