Prof. Dr. Manfred Schmidt-Schauß
Künstliche Intelligenz/Softwaretechnologie

Fachbereich Informatik und Mathematik/ Institut für Informatik
Goethe-Universität Frankfurt am Main

Grundlagen der Programmierung 2

Sommersemester 2017

Aufgabenblatt Nr. 7

Abgabe: Mittwoch 7. Juni 2017 **vor!** der Vorlesung

Aufgabe 1 (35 Punkte)

Wie in der Vorlesung und im Skript erläutert kann die Auswertung einer Stackmaschine mit den Befehlen `pop`, `pushK`, `push`, `slide`, `branchz`, `jump` und mit arithmetischen Operationen $+$, $-$, und $*$ sowie Marken wie folgt als Funktion I definiert werden:

Programm	Stack	Kopie	Nächster Aufruf (Resultat)
I []	*stack*	*prg*	$\longrightarrow$ *stack*
I (`pop` : *prest*)	(a : *srest*)	*prg*	$\longrightarrow$ I *prest srest prg*
I ((`pushK` k) : *prest*)	*stack*	*prg*	$\longrightarrow$ I *prest* (k : *stack*) *prg*
I ((`push` i) : *prest*)	*stack*	*prg*	$\longrightarrow$ I *prest* ((*stack* `!!` i) : *stack*) *prg*
I ($+$: *prest*)	(a : b : *srest*)	*prg*	$\longrightarrow$ I *prest* ($b + a$: *srest*) *prg*
I ($-$: *prest*)	(a : b : *srest*)	*prg*	$\longrightarrow$ I *prest* ($b - a$: *srest*) *prg*
I ($*$: *prest*)	(a : b : *srest*)	*prg*	$\longrightarrow$ I *prest* ($b * a$: *srest*) *prg*
I ((`slide` m n) : *prest*)	*stack*	*prg*	$\longrightarrow$ I *prest* (`take` m *stack* `++` (`drop` ($n + m$) *stack*)) *prg*
I (*marke* : *prest*)	*stack*	*prg*	$\longrightarrow$ I *prest stack prg*
I ((`branchz` *marke*) : *prest*)	(a : *srest*)	*prg*	$\longrightarrow$ `if` ($0 == a$) `then` I (`dropWhile` (*marke* `/=`) *prg*) *srest prg* `else` I *prest srest prg*
I (`jump` *marke*) : *prest*)	*stack*	*prg*	$\longrightarrow$ I (`dropWhile` (*marke* `/=`) *prg*) *stack prg*

a) Führen Sie das Stackmaschinenprogramm (bestehend aus 15 Befehlen)

```
1  pushK 13;        6  slide 1 1;       11 +;
2  pushK 2;         7  marke1;          12 -;
3  pushK 4;         8  branchz marke2;  13 jump marke1;
4  *;               9  pushK 7;         14 marke2;
5  push  1;         10 pushK 6;         15 pushK 50;
```

beginnend mit leerem Stack per Hand entsprechend der obigen Regeln aus, indem Sie das Programm und den Stack nach Ausführung jedes Befehls angeben. (10 Punkte)

b) In dieser Aufgabe soll die Stackmaschine in Haskell implementiert werden. Wir verwenden dafür den folgenden Datentyp `StackCmd`, der die Befehle der Maschine repräsentiert. Für Marken werden dabei Zahlen verwendet und die arithmetischen Operationen werden zusammengefasst durch den Datentyp `PrimOp`:

```
data StackCmd =   PushK Int | Pop | Push Int | Op PrimOp | Mark Marke
                | Jump Marke | Branchz Marke | Slide Int Int
  deriving (Eq,Show)

type Marke = Int

data PrimOp = Add | Mult | Sub | Div
 deriving(Eq,Show)
```

Ein Stackmaschinenprogramm ist eine Liste von Befehlen und der Stack ist eine Liste von Zahlen:

```
type StackProgram = [StackCmd]
type Stack = [Int]
```

Implementieren Sie in Haskell eine Funktion `runStackProgram :: StackProgram -> Stack`, die ein Stackmaschinenprogramm erwartet und den resultierenden Stack als Ergebnis liefert. Implementieren Sie hierfür zunächst eine Hilfsfunktion

```
interpret :: StackProgram -> Stack -> StackProgram -> Stack
```

die genau die oben angegebene Funktion I in Haskell umsetzt und rufen Sie diese innerhalb von `runStackProgram` auf. (25 Punkte)

Aufgabe 2 (65 Punkte)

In dieser Aufgabe betrachten wir das Spiel *1D-Pong*, das ein Variante des Spiels *Pong* im eindimensionalen Raum ist. Das Spielfeld wird durch den Stack modelliert: $[1, i, j, k, 2]$ mit $i, j, k \in \{0, 3\}$, so dass genau einer der drei Werte i, j, k 3 ist und die restlichen beiden jeweils 0 sind. Dabei repräsentiert 0 ein leeres Feld, während sich bei einer 3 der Ball auf dem Feld befindet. Die Zahl 1 repräsentiert den ersten und die Zahl 2 den zweiten Spieler.

Zur Lösung der folgenden Aufgaben kann der Stackmaschinen Simulator nützlich sein:

`http://www.ki.informatik.uni-frankfurt.de/stackmachine`

a) Angenommen der Ball befindet sich direkt beim ersten Spieler, das heißt $[1, 3, 0, 0, 2]$. Implementieren Sie ein Stackprogramm, das den Ball vom Spieler ein Feld weit weg schlägt, das heißt es soll der Stack $[1, 0, 3, 0, 2]$ zurückgeliefert werden. (5 Punkte)

b) Implementieren Sie ein Stackprogramm, das den Stack [1] zurückliefert, falls sich der Ball in der Mitte befindet, andernfalls soll [0] als Ergebnis berechnet werden. Sie können annehmen, dass der Stack anfangs von der Form $[1, i, j, k, 2]$ ist. (20 Punkte)

c) Implementieren Sie ein Stackprogramm, das den aktuellen Zustand erkennt und den Folgezustand berechnet. Falls sich der Ball gerade in der Mitte befindet, soll er in Richtung Spieler 2 fliegen. Sie können annehmen, dass der Stack anfangs von der Form $[1, i, j, k, 2]$ ist. (30 Punkte)

d) Angenommen die beiden Spieler seien perfekte Computerspieler und Spieler 1 schlägt auf. Implementieren Sie ein Stackprogramm, das ihr perfektes und somit endloses Spiel endlos durchlaufen lässt. Damit besser sichtbar ist, wann ein Zustandswechsel abgeschlossen ist, soll nach Abschluss eines jeden Zustandswechsels eine -1 oben auf den Stack gelegt werden. Für diese Teilaufgabe ist die schrittweise Ausführung mit dem Stackmaschinen Simulator zum Testen sinnvoll. (10 Punkte)