Technische Universität Berlin **WiSe 2022/23**
Fakultät II, Institut für Mathematik
Sekretariat MA 6–2, Antje Schulz
Prof. Dr. Michael Joswig
Dr. Frank Lutz, Martin Knaack, Marcel Wack

7. Programmieraufgabe Computerorientierte Mathematik I

Abgabe: 06.01.2023 über den Comajudge bis 17 Uhr

Hinweis: Die Vorstellung einer der Programmieraufgaben 04–06 ist bis zum 13.01.2023 fällig.

1 Weihnachten und Zombie-Elfen

Es ist Dezember und weihnachtet sehr. Dieses Jahr will der Weihnachtsmann seinen Elfen eine Verschnaufpause gönnen. Hierzu greift er auf etwas ungewöhnliche Hilfsarbeiter zurück: ZOMBIES! Diese hat er mit Hilfe magischer Harfen in Weihnachtselfenzombies verwandelt.
Leider ist dem Zauber nicht ganz zu trauen. Ein bestimmter Anteil der Zombies ist trotz gewissenhaftester Bezirzungen Zombie geblieben. Um diesen Umstand auszugleichen und Weihnachten vor der Zombie-Apokalypse zu bewahren, hat der Weihnachtsmann die Säcke ALLER Zombies randvoll mit magischen Harfen gefüllt. So will er die Menschen mit Musik erfreuen und ihnen darüber hinaus ein Werkzeug an die Hand geben, mit dessen Hilfe sich die Herzen heranschlurfender Zombies doch noch zum Elfentum bekehren lassen.

2 Aufgabenstellung und Anforderungen

Ihre Aufgabe ist es, eine Reihe von Hilfsfunktionen bereitzustellen, mit Hilfe derer sich das Aufeinandertreffen von Menschen, Zombies und Elfen-Zombies simulieren lässt. Die Simulation findet auf einem Rechteck statt, welches die Welt darstellt. Das Rechteck ist in Felder unterteilt, die nach dem folgenden Schema durchnummeriert sind:

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

Abbildung 1: Welt-Rechteck mit $n = 3$ und $m = 5$.

Das Welt-Rechteck wird nicht explizit angelegt. Es werden lediglich die Anzahl der Reihen n und die Anzahl der Spalten m übergeben.
Des Weiteren gibt es eine Liste `positions`, deren Einträge Listen der Länge zwei sind. Jede dieser Listen gibt Typ und Position einer Figur auf dem Rechteck an. Der erste Eintrag enthält den Typ der Figur, der zweite ihre Position. Es gibt vier Typen von Figuren, welche jeweils durch einen String identifiziert werden:

- 'Z': Zombie,
- 'ZH': Harfen verteilender Elfen-Zombie,
- 'H': Mensch (human),
- 'HH': Mensch mit Harfe, der Zombies zu Elfen-Zombies verwandeln kann.

Im folgenden Beispiel sind die vier Ecken des obigen Rechtecks mit den vier verschiedenen Figurentypen belegt:

```
positions = [['Z',0],['ZH',4],['H',10],['HH',14]]
```

Es sollen folgende Hilfsfunktionen zur Manipulation von `positions` realisiert werden:

1. Die Funktion

   `updatePosition(n,m,pos,rnd)`

   mit den Parametern

   - `n`: Anzahl der Reihen des Welt-Rechtecks,
   - `m`: Anzahl der Spalten des Welt-Rechtecks,
   - `pos`: Position der Figur als nichtnegative ganze Zahl,
   - `rnd`: zufälliger Float im halboffenen Intervall $[0, 1)$,

   soll eine aktualisierte Position der Figur als nichtnegative ganze Zahl zurückgeben. Die Richtung in die sich eine Figur bewegt, wird folgendermaßen bestimmt:

- wenn `rnd` $\in [0, 0.25)$, dann geht die Figur nach rechts,
- wenn `rnd` $\in [0.25, 0.5)$, dann geht die Figur nach links,
- wenn `rnd` $\in [0, 5, 0.75)$, dann geht die Figur nach unten,
- wenn `rnd` $\in [0.75, 1)$, dann geht die Figur nach oben.

Da das Rechteck ein Welt-Rechteck ist, kommt eine Figur die nach oben bzw. unten aus dem Rechteck läuft in der selben Spalte unten bzw. oben wieder zurück in das Rechteck. Analog kommt eine Figur die rechts bzw. links aus dem Rechteck läuft in der selben Reihe links bzw. rechts wieder zurück. Zur

Anmerkung: In der fertigen Simulation werden für `rnd` Zufallszahlen eingesetzt. Sie können hierzu `random` importieren und den Befehl `random.random()` nutzen.

Beispielaufrufe: Die folgenden Aufrufe realisieren Bewegungen in dem Welt-Rechteck aus Abbildung 1.

```
>>> updatePosition(3,5,0,0.3)
4
>>> updatePosition(3,5,4,0.8)
14
>>> updatePosition(3,5,7,0.8)
2
```

2. Die Funktion

   `updatePositions(n,m,positions)`

   führt `updatePosition` für alle Elemente in der übergebenen Liste `positions` durch. Als Parameter `rnd` wird jeweils eine Zufallszahl im Intervall $[0, 1)$ an `updatePosition` übergeben. *Die Funktion hat keinen Rückgabewert.*

3. Die Funktion

   `sortPositions(positions)`

   sortiert die übergebene Liste `positions` aufsteigend nach den zweiten Einträgen der Teillisten. *Die Funktion hat keinen Rückgabewert.*

   Beispielaufruf:

```
>>> positions = [['Z', 184], ['Z', 161], ['Z', 160], ['Z', 160]]
>>> sortPositions(positions)
>>> print(positions)
[['Z', 160], ['Z', 160], ['Z', 161], ['Z', 184]]
```

4. Die Funktion

   `extractSquare(positions)`

gibt alle Figuren auf dem Feld mit dem höchsten Index in `positions` zurück und entfernt diese aus `positions`. Dies wird realisiert, indem aus einer geordneten Liste `positions` alle letzten Elemente mit dem gleichen Eintrag in der zweiten Position entfernt und zu einer neuen Liste `square` zusammengefasst werden. Sowohl die Liste `square` als auch die veränderte Liste `positions` sollen zurückgegeben werden.

Beispielaufrufe:

```
>>> positions = [['Z', 160], ['Z', 160], ['Z', 161]]
>>> positions, square = extractSquare(positions)
>>> print(positions)
[['Z', 160], ['Z', 160]]
>>> print(square)
[['Z', 161]]
```

5. Die Funktion

   `giftExchange(square)`

   realisiert die Begegnung von Figuren auf dem gleichen Feld nach folgenden Regeln in der angegebenen Reihenfolge:

   a) Sind mindestens ein Harfen verteilender Elfen-Zombie auf dem Feld und mindestens ein Mensch, so werden alle Menschen 'H' in einen Menschen mit Harfe 'HH' umgewandelt.

   b) Sind mindestens ein Zombie 'Z' und ein Mensch ('H' oder 'HH') auf dem Feld, so wird Folgendes durchgeführt:

   - Ist die Zahl der Zombies 'Z' (die keine Harfen verteilen) $\geq$ $2\times$ der Zahl der Menschen mit Harfe, so werden alle Menschen (mit oder ohne Harfe, 'H' oder 'HH') in Zombies 'Z' umgewandelt.
   - Ist die Zahl der Zombies 'Z' (die keine Harfen verteilen) $<$ $2\times$ der Zahl der Menschen mit Harfe, so werden alle Zombies 'Z' in Harfen verteilende Zombies 'ZH' verwandelt.

   Achtung: Dieser Schritt erfolgt NACH Schritt a).

   Beispielaufrufe:

```
>>> square = [['Z', 160], ['H', 160], ['Z', 160], ['ZH', 160]]
>>> giftExchange(square)
>>> print(square)
[['Z', 160], ['Z', 160], ['Z', 160], ['ZH', 160]]
>>> square = [['H', 160], ['H', 160], ['Z', 160], ['ZH', 160]]
>>> giftExchange(square)
>>> print(square)
[['HH', 160], ['HH', 160], ['ZH', 160], ['ZH', 160]]
```

6. Die Funktion

`mergeSquare(square,intermediate)`

fügt `square` an eine gegebene Liste `intermediate` an und gibt diese zurück. Der Grundgedanke dabei ist folgender: Extrahiere so oft ein `square` aus `positions`, bis letztere Liste leer ist. Füge die square's jeweils einer Zwischenspeicher-Liste `intermediate` an und setze die Zwischenspeicher-Liste als neue `positions`-Liste, sobald `positions` leer ist.

Beispielaufruf:

```
>>> intermediate = [['HH', 160], ['HH', 160], ['ZH', 160], ['ZH', 160]]
>>> intermediate = mergeSquare([['Z', 159], ['Z', 159]],intermediate)
>>> print(intermediate)
[['HH', 160], ['HH', 160], ['ZH', 160], ['ZH', 160], ['Z', 159], ['Z', 159]]
```

7. Die Funktion

   `christmasFated(positions)`

   gibt `True` zurück, falls einer der folgenden zwei Fälle eintritt:

   - Es gibt nur noch Zombies 'Z' oder 'ZH' in `positions`.
   - Es gibt keine Zombies 'Z' mehr in `positions`.

   Ansonsten gibt die Funktion `False` zurück.

   Beispielaufrufe:

   ```
   >>> print(christmasFated([['HH', 160], ['HH', 160], ['ZH', 160], ['ZH', 160]]))
   True
   >>> print(christmasFated([['HH', 160], ['HH', 160], ['Z', 160], ['ZH', 160]]))
   False
   ```

8. Die Funktion

   `christmasFate(positions)`

   nimmt eine Liste `positions` entgegen, für welche `christmasFated` den Wert `True` ausgibt und gibt einen der folgenden Strings per `return` zurück:

   - `'Zombies ate my Christmas!'` – Falls es nur noch Zombies 'Z' oder 'ZH' in `positions` gibt.
   - `'Ho, ho, ho, and a merry Zombie-Christmas!'` – Falls es keine Zombies 'Z' und mindestens einen Menschen 'H' oder 'HH' in `positions` gibt.

   Beispielaufrufe:

   ```
   >>> christmasFate([['HH', 160], ['HH', 160], ['ZH', 160], ['ZH', 160]])
   'Ho, ho, ho, and a merry Zombie-Christmas!'
   >>> christmasFate([['Z', 160], ['Z', 160], ['ZH', 160], ['ZH', 160]])
   'Zombies ate my Christmas!'
   ```

Algorithm 1 Christmas simulation

```
1: procedure zombieChristmas(n,m,positions)
2:     while Christmas fate undecided do
3:         update positions
4:         exchange gifts
5:     end while
6:     print Christmas fate
7: end procedure
```

Die oben aufgelisteten Hilfsfunktionen sollen in einer Funktion

```
zombieChristmas(n,m,positions)
```

zu einer lauffähigen Simulation zusammengefasst werden. Hierbei sind `n,m` die Reihen- bzw. Spaltenzahl des Welt-Rechtecks und `positions` ist die initiale Liste mit den Figuren und ihren Positionen zu Beginn der Simulation. Die Simulation soll folgendem Pseudocode entsprechen: