6. Programmieraufgabe Computerorientierte Mathematik I

Abgabe: 08.01.2021 über den Comajudge bis 17 Uhr

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

Das Welt-Rechteck wird nicht explizit angelegt. Es gibt 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. `updatePosition(n,m,pos,rnd)` Nimmt folgende Parameter engegen:
   - `n`: Reihen-Zahl des Welt-Rechtecks,
   - `m`: Spalten-Zahl des Welt-Rechtecks,
   - `pos`: Position der Figur als nichtnegative ganze Zahl,
   - `rnd`: Float im halboffenen Intervall [0,1).

   Die Funktion gibt eine aktualisierte Position der Figur als nichtnegative ganze Zahl zurück. Hierbei gelten folgende Bewegungsregeln.
   - `rnd` $\in$ [0,0.25): Figur geht nach rechts,
   - `rnd` $\in$ [0.25,0.5): Figur geht nach links,
   - `rnd` $\in$ [0,5,0.75): Figur geht nach unten,
   - `rnd` $\in$ [0.75,1): Figur geht nach oben.

   Da das Rechteck ein Welt-Rechteck ist, kommt eine Figur die nach oben/unten aus dem Rechteck läuft in der selben Spalte unten/oben wieder zurück in das Rechteck. Für den Fall links/rechts gilt die analoge Regel.

   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 obigem Rechteck.

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

2. `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. `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. `extractSquare(positions)` Gibt alle Figuren auf dem Feld mit dem höchsten Index in `positions` zurück. 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.
   Beispielaufrufe:

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

5. `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 und ein Mensch 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. `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
```

7. `mergeSquare(square,intermediate)` Fügt `square` an eine gegebe Liste an. 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 an und setze die Zwischenspeicher-Liste als neue `positions`-Liste, sobald `positions` leer ist.
   Beispielaufruf:

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

8. `christmasFate(positions)` Nimmt eine Liste `positions` entgegen, für welche `christmasFated` den Wert `True` ausgibt und gibt einen der folgenden Strings (als `return`-Wert) 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 nur noch Menschen 'H' oder 'HH' und Harfen verteilende Zombies 'ZH' 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!'
```

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 sollte folgendem Pseudocode entsprechen:

Algorithm 1 Christmas simulation

```
procedure zombieChristmas(n,m,positions)
    while Christmas fate undecided do
        update positions
        exchange gifts
    end while
    print Christmas fate
end procedure
```

3 Abgabe und Vorstellung

Weil die Funktionen teils randomisiert sind, werden lediglich die deterministischen Teilfuntionen vom `comajudge` auf ihre Korrektheit hin überüft. Bei der Vorstellung jedoch soll eine lauffähige Simulation nach obigen Maßgaben demonstriert werden.