Aufgabenblatt 8

Allgemeine Informationen zum Aufgabenblatt:

- Die Abgabe erfolgt in TUWEL. Bitte laden Sie Ihre Python-Datei bis spätestens **Freitag, 26.06.2020 20:00 Uhr** in TUWEL hoch.
- Beachten Sie bitte folgende Punkte:
  - Ihre Programme müssen ausführbar sein, d.h. einzelne Programme sollten z.B. keine Syntaxfehler oder Typfehler enthalten.
  - Bei kleinen logischen Fehlern (falsche Ergebnisse) werden wir keine Punkte abziehen. Daher sollten Sie immer versuchen, alle Aufgaben vollständig abzugeben.
  - Die Angabedatei muss nicht umbenannt werden.

In diesem Aufgabenblatt werden folgende Themen behandelt:

- O-Notation
- Stack
- Queue
- Listen
- Klassen

Aufgabe 1 (1 Punkt)

Wir gehen von einem Stack aus, auf dem eine Sequenz von `push`- und `pop`-Operationen durchgeführt wird. Die `push`-Operationen legen die Zahlen von 0 bis 5 (in dieser Reihenfolge) auf den Stack. Dabei werden sie von `pop`-Operationen unterbrochen, deren Rückgabewerte in einer Zeile ausgegeben werden. Zum Beispiel sei folgende Sequenz von Operationen gegeben:

```
push(0), push(1), push(2), pop(), push(3), push(4), pop(), pop(), push(5),
pop(), pop(), pop()
```

Dann lautet die Ausgabe:

```
2 4 3 5 1 0
```

Welche der folgenden Ausgaben können nicht durch vermischte `push`- und `pop`-Operationen erzeugt werden:

- 2 1 0 3 5 4
- 2 3 5 4 0 1
- 3 2 1 4 0 5
- 0 3 1 2 5 4

Geben Sie bei nicht möglichen Ausgaben eine kurze Erklärung, warum diese nicht entstehen können.

Aufgabe 2 (1 Punkt)

Wir gehen von einer Queue aus, auf die eine Sequenz von `enqueue`- und `dequeue`-Operationen durchgeführt wird. Die `enqueue`-Operationen geben die Zahlen von 0 bis 5 (in dieser Reihenfolge) in die Queue. Dabei werden sie von `dequeue`-Operationen unterbrochen, deren Rückgabewerte in einer Zeile ausgegeben werden. Zum Beispiel sei folgende Sequenz von Operationen gegeben:

```
enqueue(0), enqueue(1), enqueue(2), dequeue(), enqueue(3), enqueue(4),
dequeue(), dequeue(), enqueue(5), dequeue(), dequeue(), dequeue()
```

Dann lautet die Ausgabe:

```
0 1 2 3 4 5
```

Welche der folgenden Ausgaben können nicht durch vermischte `enqueue`- und `dequeue`-Operationen erzeugt werden:

- 5 4 3 2 1 0
- 5 4 0 1 2 3
- 2 1 0 3 4 5
- 0 1 2 3 5 4

Geben Sie bei nicht möglichen Ausgaben eine kurze Erklärung, warum diese nicht entstehen können.

Aufgabe 3 (1 Punkt)

Implementieren Sie eine Funktion `sort_stack`, die einen Stack übergeben bekommt und dessen Inhalt in einem neuen Stack aufsteigend sortiert zurückgibt. Das größte Element steht danach an der Position, die von der Stack-Operation `top` zurückgeliefert wird. Folgende Punkte müssen dabei beachtet werden:

- Sie dürfen keine Annahmen über die Implementierung des Stacks treffen.
- Sie dürfen nur die Stack-Operationen `push`, `pop`, `top` und `is_empty` benutzen.

Sie haben in der Angabedatei eine Testfunktion gegeben. Beim Ausführen muss die Ausgabe folgendermaßen lauten:

```
[8, 7, 6, 5, 4, 3, 2, 1]
```

Beachten Sie weiters:

- Weil der Stack von der obersten Stelle weg ausgelesen wird, sind die Zahlen absteigend sortiert.
- Sie können die Stack-Implementierung aus der Vorlesung verwenden (import ist schon vorgesehen).
- Geben Sie im zusätzlichen Kommentar die kleinste obere Schranke für die Laufzeit Ihrer Implementierung an.
- Sie müssen nur die implementierte Funktion zusammen mit der Testfunktion abgeben.

Aufgabe 4 (1 Punkt)

Implementieren Sie eine Funktion `reverse_first_k_entries(k, queue)`, die die ersten k Einträge einer Queue `queue` in umgekehrte Reihenfolge bringt. Sei zum Beispiel $k = 4$ und die Queue hat folgenden Inhalt:

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

Dann ist der Inhalt nach dem Aufruf der Funktion:

```
[40, 30, 20, 10, 50, 60, 70, 80, 90]
```

Sie müssen folgende Punkte beachten:

- Sie dürfen nur die Operationen `enqueue`, `dequeue` und `len(queue)` bei der Queue verwenden.
- Sie dürfen einen zusätzlichen Stack (`push`, `pop` und `is_empty` erlaubt) verwenden.
- Die Funktion sollte nur dann eine Änderung vornehmen, wenn $1 < k <= len(queue)$ gilt. Ansonsten wird nichts verändert.

Beachten Sie weiters:

- Sie können die Stack- und Queue-Implementierungen aus der Vorlesung verwenden (import ist schon vorgesehen).
- Geben Sie im zusätzlichen Kommentar die kleinste obere Schranke für die Laufzeit Ihrer Implementierung an.
- Sie müssen nur die implementierte Funktion zusammen mit der Testfunktion abgeben.

Aufgabe 5 (1 Punkt)

Sie haben eine Implementierung einer speziellen Liste gegeben. Beantworten Sie folgende Fragen:

- An welcher Stelle wird ein neuer Knoten eingefügt?
- Was macht die Methode `flip`? Beschreiben Sie kurz, wie die Methode `flip` abläuft und ihr Ergebnis erzeugt. Konkret geht es darum, wie viele Positionen in der Liste sich die Methode merkt und wie Knoten an einzelnen Positionen miteinander verbunden werden.

Tipp: Zeichnen Sie sich eine kleine Liste (z.B. mit 3-4 Einträgen) auf und gehen Sie die Methode `flip` Zeile für Zeile durch.

Aufgabe 6 (2 Punkte)

Sie haben eine Implementierung einer speziellen Liste gegeben. Beantworten Sie folgende Fragen:

- Wie ist diese Liste aufgebaut? Welche Knoten sind schon automatisch vorhanden?
- An welcher Stelle wird ein neuer Knoten eingefügt?

Ergänzen Sie weiters die Methode `__str__` so, dass die Testfunktion `test_tangled_list()` folgende Ausgabe produziert:

```
[]
[1]
[1, 2, 3]
```

Aufgabe 7 (3 Punkte)

Implementieren Sie zunächst eine Klasse `Song`, die folgende Datenattribute enthält:

- `title` speichert den Titel eines Liedes.
- `artists` speichert den Namen der Person bzw. Gruppe, die für das Lied verantwortlich ist.
- `play_count` speichert, wie oft ein Lied abgespielt wurde.

Die Klasse `Song` sollte folgende Methoden anbieten:

- Mit `__init__` werden die drei Datenattribute mit Werten belegt. Dabei werden `title` und `artists` als Argumente übergeben. `play_count` wird in der Methode zunächst auf 0 gesetzt.
- Mit `play` wird das Lied abgespielt. Das wird in unserem Programm nur simuliert, d.h. es wird mit einer Ausgabe auf der Konsole gezeigt, dass das Lied abgespielt wird und `play_count` wird um 1 erhöht. Wurde das Lied schon 5 mal gespielt, dann wird der Zähler nicht mehr erhöht. Beispiel für Ausgabe: Playing * Let It Be * by Beatles
- Mit `__str__` wird der Inhalt eines Objekts in der Form `title - artists (play count = x)` ausgegeben, wobei `x` der Anzahl entspricht, die bei einem Lied gespeichert wird. Beispiel: Let It Be - Beatles (play count = 0)

Implementieren Sie dann eine Klasse `Player`, die als Datenattribut eine Liste von Liedern hat. Folgende Methoden sollen implementiert werden:

- Mit `__init__` werden die Lieder aus der übergebenen Liste in einer Deque gespeichert. Sie dürfen hier auf die Implementierung einer Deque in Python zurückgreifen. Sie können davon ausgehen, dass eine Liste mit zumindest einem Lied übergeben wird und brauchen keine Überprüfung auf eine leere Liste implementieren.
- Mit `play_first_song` wird das erste Lied aus der Deque abgespielt. Dazu wird das Lied aus der Deque entfernt, die Methode `play` darauf aufgerufen und danach das Lied wieder am Ende der Deque eingefügt. Sie können einen zusätzlichen Text ausgeben (siehe Beispielausgabe am Ende der Aufgabe).
- Mit `play_last_song` wird das letzte Lied aus der Deque abgespielt. Dazu wird das Lied aus der Deque entfernt, die Methode `play` darauf aufgerufen und danach das Lied wieder am Anfang der Deque eingefügt. Sie können einen zusätzlichen Text ausgeben (siehe Beispielausgabe am Ende der Aufgabe).
- Mit `__str__` wird der Inhalt eines Players ausgegeben. Dazu werden die einzelnen Lieder ausgegeben (ein Lied pro Zeile). Die Formatierung (zusätzlicher Text) bleibt Ihnen überlassen (siehe Beispielausgabe am Ende der Aufgabe).

Implementieren Sie einen Test für diese beiden Klassen:

- Legen Sie zumindest 5 Lieder an.
- Erzeugen Sie dann ein Player-Objekt.
- Geben Sie den Inhalt des Player-Objekts aus.
- Spielen Sie dann mehrmals das erste bzw. das letzte Lied in der Deque ab. Zumindest ein Lied sollte mehr als 5 mal abgespielt werden.
- Geben Sie den Inhalt des Player-Objekts aus.

Nachfolgend ein Beispiel für eine mögliche Ausgabe:

```
Player list:
Let It Be - Beatles (play count = 0)
Nowhere Man - Beatles (play count = 0)
The Sound of Silence - Simon and Garfunkel (play count = 0)
Blowin' in the Wind - Bob Dylan (play count = 0)
Piano Man - Billy Joel (play count = 0)

Play first song:
Playing * Let It Be * by Beatles
Play first song:
Playing * Nowhere Man * by Beatles
Play first song:
Playing * The Sound of Silence * by Simon and Garfunkel
Play first song:
```

```
Playing * Blowin' in the Wind * by Bob Dylan
Play first song:
Playing * Piano Man * by Billy Joel
Play first song:
Playing * Let It Be * by Beatles
Play first song:
Playing * Nowhere Man * by Beatles
Play first song:
Playing * The Sound of Silence * by Simon and Garfunkel
Play first song:
Playing * Blowin' in the Wind * by Bob Dylan
Play first song:
Playing * Piano Man * by Billy Joel
Play first song:
Playing * Let It Be * by Beatles
Play last song:
Playing * Let It Be * by Beatles
Play first song:
Playing * Let It Be * by Beatles
Play last song:
Playing * Let It Be * by Beatles
Play first song:
Playing * Let It Be * by Beatles
Play last song:
Playing * Let It Be * by Beatles

Player list:
Let It Be - Beatles (play count = 5)
Nowhere Man - Beatles (play count = 2)
The Sound of Silence - Simon and Garfunkel (play count = 2)
Blowin' in the Wind - Bob Dylan (play count = 2)
Piano Man - Billy Joel (play count = 2)
```