Programmierkurs Python I – WS 09/10

Übung 3

---

1 Primzahlen mit `for` (3 Punkte)

In der 2. Vorlesung wurde ein Algorithmus vorgestellt, der die Primzahlen in einem bestimmten Zahlenbereich berechnet und dafür eine `while`-Schleife benutzt.

Schreibe eine Funktion, die Primzahlen berechnet und dafür eine `for`-Schleife benutzt. Die Funktion soll als Parameter die Grenzen des Zahlenbereichs nehmen. Rückgabewert soll eine Liste der entsprechenden Primzahlen sein.

```
>>> primzahlen(2,50)
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

2 Quicksort (4 Punkte)

Ein rekursiver Sortieralgorithmus für Listen ist Quicksort:

- Wähle ein zufälliges Element `e` aus der Liste (das „Pivotelement“)
- Erstelle zwei Listen; in der ersten Liste stehen alle Elemente, die kleiner als `e`, in der zweiten Liste alle Elemente, die größer oder gleich `e` sind.[1]
- Sortiere die beiden neuen Listen.
- Gib eine Liste zurück, in der zuerst die erste Liste, dann `e` und zuletzt die zweite Liste aufeinander folgen.

Implementiere eine Quicksort-Funktion, die als Parameter eine Liste mit Zahlen nimmt und eine sortierte Liste zurück gibt. (Benutze *nicht* Pythons `sort`-Funktion!)

```
>>> quicksort([5, 1, 23, 934, 42, 3234, 432, 234, 561, 451, 4, 5, 1, 123, 54])
[1, 1, 4, 5, 5, 23, 42, 54, 123, 234, 432, 451, 561, 934, 3234]
```

[1] `e` selbst ist in keiner der Listen. Der ursprüngliche Algorithmus sieht vor, dass die Elemente, die gleich `e` sind, zufällig auf die beiden Listen verteilt werden können; wir vereinfachen das hier.

3 Fibonacci iterativ (3 Punkte)

Auf den Folien zur 3. Vorlesung wird ein rekursiver Algorithmus zur Berechnung der Fibonacci-Zahlen gezeigt. Implementiere eine *nicht-rekursive* Funktion, die die Fibonacci-Zahlen mit einer Schleife berechnet.

```
>>> fibonacci_iterativ(5)
5
```

4 Terminkalender (3 + 1 Punkte)

Implementiere zwei Funktionen, die zusammen eine (sehr einfache) Terminkalender-Funktionalität liefern. Die erste Funktion `addEvent(year,month,day,what,cal)` soll einen Termin zu einem Kalender hinzufügen. Parameter sollen Jahr, Monat und Tag (als Zahlen), ein String für den Termin-Inhalt und der Kalender sein.

Die zweite Funktion `getEvents(year,month,day,cal)` soll aus dem Kalender alle Termine für das angegebene Datum (Format wie oben) heraussuchen und als sinnvoll formatierten String zurückgeben.

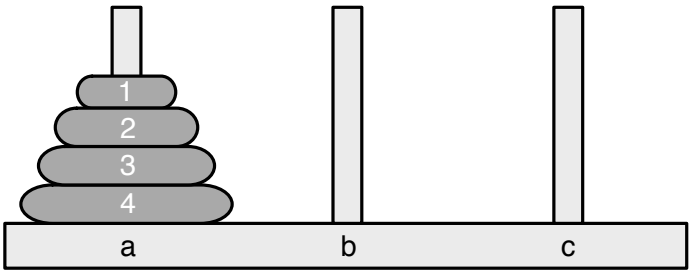
Wähle sinnvolle Datenstrukturen für den Terminkalender. Ein möglicher Programmablauf könnte so aussehen:

```
>>> cal = ... % ein leerer Kalender, mit Eurer Struktur initialisiert
>>> addEvent(2009,11,12,"Python I, VL", cal)
>>> addEvent(2009,11,12, "CoLi Kolloquium", cal)
>>> addEvent(2009,12,24, "Weihnachten",cal)
>>> print(getEvent(2009,11,12,cal))
['Python I, VL', 'CoLi Kolloquium']
```

Bonusaufgabe: Implementiere zwei zusätzliche Methoden `getEventsForMonth(year,month,cal)` und `getEventsForYear(year,cal)`, die alle Termine für einen gegebenen Monat bzw. ein gegebenes Jahr als String zurückliefern.

```
>>> print(getEventsForMonth(2009,11,cal))
12.: ['Python I, VL', 'CoLi Kolloquium']
>>> print(getEventsForYear(2009,cal))
12.11.: ['Python I, VL', 'CoLi Kolloquium']
24.12.: ['Weihnachten']
```

5 Die Türme von Hanoi (0 + 3 Punkte)



Die Türme von Hanoi sind ein berühmtes Beispiel für Probleme, die elegant rekursiv gelöst werden können. Der Spielaufbau besteht aus drei Stäben (im Bild a, b, c) und n Scheiben (im Bild n=4). Ziel ist es, den Scheibenturm von a auf einen anderen Stab zu bringen (z.B. c). Man darf jeweils nur eine Scheibe bewegen, und eine Scheibe darf nur auf eine größere Scheibe oder einen leeren Stab gelegt werden.

Bonusaufgabe: Finde und implementiere den Algorithmus zur Lösung des Problems für n Scheiben. Eine Möglichkeit zur Implementierung besteht darin, die Spielzüge in der richtigen Reihenfolge auszugeben (z.B. *Bewege Scheibe x von a nach c.*).

Abgabe bis Donnerstag, 19.11.09, 14.00 Uhr
per Mail an
`regneri@coli.uni-sb.de`
`lcarolyn@coli.uni-sb.de`