ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Dr. Jochen Hoenicke
Alexander Nutz

12.5.2014
Abgabe: Montag 19.5.2014
via Daphne/Subversion

3. Übungsblatt zur Vorlesung
Einführung in die Informatik

Aufgabe 1: Rekursive Funktionen

Schreiben Sie ein Programm, das für folgende Funktionen über den natürlichen Zahlen deren Parameter von der Kommandozeile einliest und darauf mit dem jeweiligen Funktionswert antwortet.

- Summe der ersten n positiven natürlichen Zahlen:

$$f(1) = 1$$
$$f(n) = n + f(n-1)$$

- Die n-te Fibonacci Zahl:

$$fib(0) = 0$$
$$fib(1) = 1$$
$$fib(n) = f(n-1) + f(n-2)$$

- Die Ackermann-Péter Funktion: (mit niedrigen Parametern testen!)

$$a(m,n) = \begin{cases} n+1 & \text{falls } m = 0 \\ a(m-1,1) & \text{falls } m > 0 \text{ und } n = 0 \\ a(m-1, a(m,n-1)) & \text{falls } m > 0 \text{ und } n > 0 \end{cases}$$

Aufgabe 2: Klassen

Schreiben Sie ein Programm, das die Daten eines Auto-Quartett-Spiels einliest. Jede Karte soll in einem Objekt gespeichert werden. Das gesamte Quartett soll in einem Array gespeichert werden.
Die Werte sind vom Benutzer einzugeben etwa in der Form:

```
Wieviele Karten soll das Spiel enthalten? <Benutzereingabe n>
<wiederhole n mal:>
Erstelle <n>te Karte.
```

```
Typbezeichung des Autos? <Benutzereingabe>
Zylinder? <Benutzereingabe>
Hubraum? <Benutzereingabe>
Feinstaub-Plakette (4 = grün, 3 = gelb, 2 = rot, 1 = keine)?
<Benutzereingabe>
```

(Sie können auch eigene Werte einführen, es muss auch kein Auto-Quartett sein.)
Am Schluss soll das ganze Spiel auf der Konsole ausgegeben werden.
(Sie können es dann zum Beispiel ausdrucken, die Karten ausschneiden und damit spielen.)

Aufgabe 3: Vier gewinnt 8 Punkte

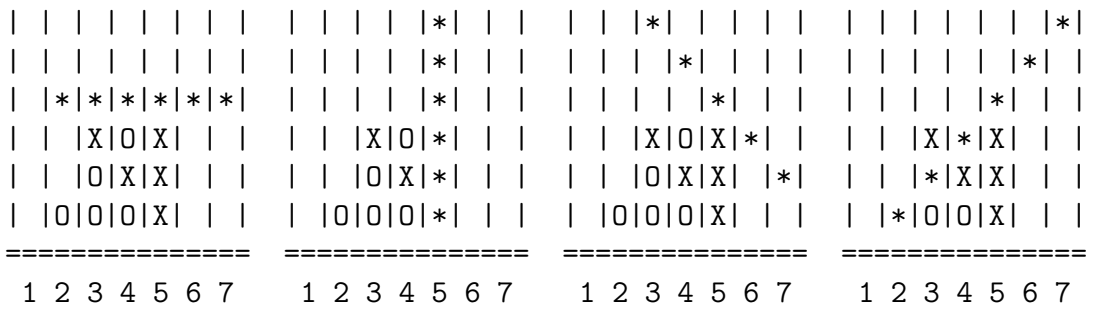
Schreiben Sie ein Vier gewinnt Spiel.
Speichern Sie dazu den jeweils aktuellen Spielzustand in einem zweidimensionalen Array. (Standard Vier gewinnt hat sechs Zeilen und sieben Spalten, Sie können auch ein größeres programmieren, oder die Größe von Parametern abhängig machen.) Nutzen Sie eine Anzeige in ASCII-Art. Die Spalten sollten durchnummeriert sein, so dass die Spieler ihren Zug durch Eingabe einer Zahl machen können. Für die Felder bietet sich an, ein `X` für einen roten Stein, ein `O` für einen blauen Stein und ein Leerzeichen für ein leeres Feld zu wählen. Zum Beispiel so:

```
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | |X|O|X| | |
| | |O|X|X| | |
| |O|O|O|X| | |
===============
 1 2 3 4 5 6 7
```

Überprüfen Sie für jede Eingabe, ob sie einen korrekten Zug darstellt. Ein korrekter Zug ist die Angabe einer existierenden Spalte, die noch nicht voll ist.
Bei einer korrekten Eingabe kommt der eingeworfene Stein auf dem obersten Feld, das in der jeweiligen Spalte noch frei ist zum liegen.
Ihr Programm soll nach jedem getätigten Zug überprüfen, ob der Spieler, der zuletzt dran war, durch seinen letzten Zug gewonnen hat. Das kann zum Beispiel geschehen, indem horizontal, vertikal, und in den beiden Diagonalen die angrenzenden Felder überprüft werden, etwa so:
Angenommen in obiger Spielsituation ist der Spieler mit den `O`s als nächstes am Zug. Und wirft einen Stein in Spalte 5. Es ergibt sich folgender Zustand:

```
| | | | | | | |
| | | | | | | |
| | | | |O| | |
| | |X|O|X| | |
| | |O|X|X| | |
| |O|O|O|X| | |
===============
 1 2 3 4 5 6 7
```

Um nun zu testen, ob der Spieler gewonnen hat, genügt es nacheinander die Horizontale, Vertikale und die zwei Diagonalen, die durch das Feld laufen, wo der letzte Stein eingeworfen wurde, zu prüfen. (und zwar nur maximal drei Felder in jede Richtung):



Wenn ein Spieler gewonnen hat oder das Spielfeld voll ist (der Ausgang ist dann Unentschieden), soll das Spiel abbrechen und eine entsprechende Meldung ausgeben.