Aufgaben zur Veranstaltung
Algorithmen und Datenstrukturen, SS 2019

H. Pflug, J. Dietel FH Aachen, Campus Jülich; IT Center, RWTH Aachen

Präsenzaufgaben 09 20./21.05.2019

Die Lösung der Aufgaben wird am Ende der Übung von Ihnen vorgestellt.

Aufgabe 1

Schreiben Sie eine Funktion

```
public static void printPermutationen(int n)
```

die alle Permutationen der Zahlenfolge 1,2,…,n auf dem Bildschirm ausgibt.

Beispiel: Die möglichen Permutationen der Zahlenfolge 1,2,3 sind:

1,2,3
1,3,2
2,1,3
2,3,1
3,1,2
3,2,1

n kann auch größer als 9 sein. Als Trennzeichen können beliebige Zeichen verwendet werden.

Aufgabe 2

Schreiben Sie eine Funktion

```
public static void printPermutationen(int n, int ges)
```

die alle Strings auf dem Bildschirm ausgibt, die die folgenden Eigenschaften erfüllen:

- Der String besteht ausschließlich aus den Zeichen 0 und 1.
- Die Länge des Strings ist `ges`.
- Der String enthält genau `n` Einsen.

Beispiel: Der Aufruf `printPermutationen(2,4)` gibt folgende Strings auf dem Bildschirm aus:

1100, 1010, 1001, 0110, 0101, 0011

Die Reihenfolge ist dabei nicht maßgeblich.

Aufgabe 3

Schreiben Sie eine Funktion, die die Lösung des Spiels „Türme von Hanoi“ ermittelt. Sehen Sie sich die Regeln des Spiels unter

https://de.wikipedia.org/wiki/Türme_von_Hanoi

an. Schreiben Sie dann eine Funktion

```
public static void printBewegungen(int n, int start, int ziel)
```

die die Bewegungen ausgibt, die nötig sind, um einen Stapel von *n* Scheiben von Stapel *start* zu Stapel *ziel* zu bewegen.

Beispiel: Der Funktionsaufruf

```
printBewegungen(3,1,2)
```

soll folgenden Text ausgeben:

```
Bewege Scheibe von 1 nach 2
Bewege Scheibe von 1 nach 3
Bewege Scheibe von 2 nach 3
Bewege Scheibe von 1 nach 2
Bewege Scheibe von 3 nach 1
Bewege Scheibe von 3 nach 2
Bewege Scheibe von 1 nach 2
```

Benutzen Sie einen rekursiven Algorithmus.

Tipp: Die Nummer des dritten Stapels kann man mit der Formel *n=6-start-ziel* berechnen.

Zusatzaufgabe (alte Klausuraufgabe)

Es gibt ein Rechenspiel, das darin besteht, in einem eindimensionalen Zahlenfeld vom ersten Element zum letzten zu gelangen, wobei das Ziel ist, dabei so wenige Punkte wie möglich zu sammeln.

Der Spieler startet am ersten Element. Er hat zwei Möglichkeiten:

- Er kann 3 Elemente nach vorne springen
- Er kann ein Element nach hinten gehen.

Ist der Spieler am letzten Element angelangt, zählt er alle Punkte der berührten Elemente zusammen und erhält seine Gesamtpunktzahl. Ziel ist es, den Weg mit der kleinsten Punktzahl zu finden.

Beispiel: Das Zahlenfeld [5, 2, 9, 2, 9, 5, 3, 10, 5] lässt sich am besten wie folgt durchlaufen:

- Start bei 5
- 3 vorwärts zur 2
- 3 vorwärts zur 3
- Rückwärts zur 5
- 3 vorwärts zur 5

Das ergibt 20 Punkte

a) Schreiben Sie eine Funktion

```
public static ArrayList<Integer> shortestWay(int [] f)
```

die die Indizes der berührten Punkte auf dem kürzesten Weg zurückgibt. Verwenden Sie einen Backtracking-Algorithmus zur Lösung des Problems.

b) Beschreiben Sie kurz, wie Sie verhindern, dass unendlich lange Wege (durch unendlich oft durchlaufene Schleifen) getestet werden und das Programm dadurch abbricht.