3. Übungsblatt : Grundlagen der Programmierung [9 Punkte]

Spielregeln:

- Achten Sie auf eine exakte Formatierung Ihrer Ausgaben.
- Sie dürfen keine Bibliotheksfunktionen benutzen, es sei denn wir weisen explizit auf eine hin.

Aufgabe 1 (Zahlensysteme) [3+3 Punkte]:

„Römische Zahlen“

Die **Römische Zahlschrift** ordnet bestimmten Buchstaben einen Wert zu und repräsentiert einen Zahlenwert durch Aneinanderreihung dieser Buchstaben. Die einzelnen Buchstabenwerte sind folgender Tabelle zu entnehmen:

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Die Buchstaben werden der Größe nach aufgelistet und ihre Werte aufsummiert. Einzige Ausnahme bildet die Subtraktionsregel, die verhindern soll, dass mehr als dreimal der gleiche Buchstabe aufeinander folgt. So wird zum Beispiel die 4 als IV geschrieben, und nicht als IIII. Die Buchstaben I, X und C dürfen dazu maximal einmal vor einem anderen Buchstaben stehen. Folgende Kombinationen sind erlaubt:

	I	X	C
darf stehen vor:	V, X	L, C	D, M

So wird die Zahl **1999** als **MCMXCIX** dargestellt (M + CM + XC + IX = 1000 + 900 + 90 + 9).

a) Schreiben Sie ein Java-Programm `Dec2Rom`, das von der Standardeingabe eine Dezimalzahl einliest und die Zahl entsprechend im Römischen Zahlensystem ausgibt. Sie dürfen zum Einlesen die Klasse **`gdp.stdlib.StdIn`** benutzen.

b) Schreiben Sie ein Java-Programm `Rom2Dec`, das von der Standardeingabe eine Zahl im Römischen Format einliest und die entsprechende Dezimalzahl ausgibt. Sie dürfen die Klasse **`gdp.stdlib.StdIn`** benutzen und insbesondere die Methode **`readChar()`**.

Hinweise:

- Die beiden Programme müssen im Wertebereich {1, 2, …, 3000} funktionieren.

- Sie können davon ausgehen, dass die Programme immer eine korrekte Eingabe erhalten. Insbesondere brauchen Sie nicht zu überprüfen, ob die Zahlenbereiche eingehalten werden.
- Für die Römische Schreibweise verwenden Sie ausschließlich Großbuchstaben.

Beispiele (real, Sie müssen die gleichen Ergebnisse erhalten):

```
$ echo 1984 | java Dec2Rom
MCMLXXXIV
$ echo MCMLXXXIV | java Rom2Dec
1984
$ echo 1984 | java Dec2Rom | java Rom2Dec
1984
```

Abgabedateien im Goya: `Dec2Rom.java` und `Rom2Dec.java`

Aufgabe 2 (Methoden, Rekursion) [3 Punkte]:

„**Münzen**“

Regulär gibt es die folgenden Euromünzen $c_0, c_1, \ldots, c_{n-1}$ (hier n = 8):

Schreiben Sie ein Java-Programm `Coins`, welches für beliebige Geldbeträge x unter 10 € (in Cent angegeben) ermittelt, auf wie viele verschiedene Arten dieser Betrag passend ausgezahlt werden kann.

Die Menge $M(x) = \{\langle x_0, x_1, \ldots, x_{n-1}\rangle | \sum_{i=0}^{n-1} x_i * c_i = x\}$ beschreibt alle möglichen Münzkombinationen für den Betrag x. Somit ist $|M(x)|$ die gesuchte Zahl.

Beobachtung: Die Menge $M(x)$ lässt sich in zwei disjunkte Teilmengen M_1 (zahle den Betrag x ohne Verwendung der Münze c_i) und M_2 (zahle den Betrag x, wobei die Münze c_{n-1} mindestens einmal vorkommt) zerlegen. Falls x kleiner ist als c_{n-1} ist M_2 leer. Außerdem gilt $|M_2(x)| = |M(x - c_{n-1})|$.

Damit ist das Problem auf die Lösung von zwei einfacheren Problemen reduziert worden. Unter Hinzunahme geeigneter Abbruchbedingungen sollte sich daraus ein rekursiver Algorithmus bauen lassen.

Hinweise:

- Sie können davon ausgehen, dass von jeder Münzsorte potenziell beliebig viele Münzen verfügbar sind.

Beispiele (real, Sie müssen die gleichen Ergebnisse erhalten):

```
$ java Coins 123
123 can be paid in 9892 ways
$ java Coins 555
555 can be paid in 11054680 ways
```

Abgabedatei im Goya: `Coins.java`

Abgabehinweise

- Die angegebenen Ausgaben zeigen nur ausgewählte Beispiele. Ihre Programme sollen für den gesamten Wertebereich korrekt funktionieren.
- Referenzrechner (dort sollte Ihre Lösung funktionieren): star.informatik.hu-berlin.de
- Die **Abgabe** der Übungsaufgaben erfolgt **in Gruppen**, ideale Gruppengröße=2.
  Wichtig: Erst Gruppe bilden, **dann Lösungen** hochladen!
- Folgende Dateien sind **rechtzeitig über Goya** als **UTF-8** kodierte Lösungsquelltexte abzugeben:
  `Dec2Rom.java`
  `Rom2Dec.java`
  `Coins.java`