A3

A Protokoll für die Klasse Menge implementieren

Schreiben Sie eine Klasse MySet, die die folgenden Methoden implementiert. Informieren Sie sich zuvor in der Dokumentation der Klasse Set in Ruby über die Funktionsweise der Methoden. Repräsentieren Sie eine Menge intern als Array.

```
class MySet

  def initialize(ary=[])
  end

  def &(other_set)
  end

  def |(other_set)
  end

  def -(other_set)
  end

  def <<(an_elem)
  end

  def ^(other_set)
  end

  def add?(an_elem)
  end

  def clear()
  end

  def delete(an_elem)
  end

  def delete?(an_elem)
  end

  def empty?()
  end

  def include?(an_elem)
  end

  def proper_subset?(other_set)
  end

  def proper_superset?(other_set)
  end

  def subset?(other_set)
  end

  def super_set?(other_set)
  end

  def to_a()
  end
end
```

B Darlehensentwicklung

Schreiben Sie eine Methode ***darlehens_entwicklung(K,j,r,t)***, das die Darlehensentwicklung für einen Darlehensbetrag K, eine Laufzeit von Jahren j, einer jährlichen Zinsrate von r und einer Tilgungsrate von t in einer Tabelle einträgt. Schreiben Sie eine Methode, die den Inhalt der Tabelle mit geeigneten Spaltennamen ausgibt.

Die Tabelle enthält für jedes Jahr und jeden Monat Einträge über den Restdarlehensbetrag, die monatlichen Zinsen, die monatliche Tilgung, sowie die bis zu dem Monat akkumulierten Zinsen.

Der Restdarlehensbetrag ergibt sich aus der Differenz des Restdarlehensbetrags des Vormonats und der für den Vormonat geleisteten Tilgung.

C Binomialkoeffizienten berechnen

Sie sollen für ein N, das Sie von der Konsole einlesen die Binomialkoeffizienten berechnen und diese in einem zweidimensionalen ungleichförmigen Array a speichern, so dass a[N][k] die Wahrscheinlichkeit enthält, dass Sie genau k –mal Kopf erhalten, wenn Sie eine Münze N mal werfen.

Die Berechnungsvorschrift, die sich aus dem Pascalschen Dreieck ableitet, lautet:

a[0][k] = 0, für alle k

a[N][k+1] = 0, für alle N, alle k

a[1][1] = 1

a[N][k] = (a[N-1][k] + a[N-1][k-1]) /2

Schreiben Sie dazu die Methode ***binomial_koeffizienten(n),*** die das Array erzeugt, sowie eine Methode ***zwei_dim_ary_ausgeben(zwei_dim_ary)***, die das Ergebnis auf der Konsole ohne die 0'te Spalte und das jeweils letzte Element einer Zeile, das (k+1)'te Element ausgibt.

Für N = 5 sieht die Ausgabe wie folgt aus:

```
1
0.5 0.5
0.25 0.5 0.25
0.125 0.375 0.375 0.125
0.0625 0.25 0.375 0.25 0.0625
```

D Matrixoperationen

Implementieren Sie die folgenden Matrixoperationen

```
class Matrix

  # Multiplikation mit einem Vektor: Skalarprodukt einer nxm Matrix mit einem
  # Spaltenvektor der Länge m. Das Ergebnis ist ein Vektor der Länge n
  def multipliziere_mit_vektor(vektor)
  end
  # Matrixmultiplikation: Multipliziert eine nxm Matrix mit eine mxk Matrix
  # das Ergebnis ist eine nxk Matrix
  # die Elemente in der nxk Matrix res[i][l] werden berechnet, indem
  # die Summe über die Produkte a[i][j]*b[j][l] für j = 0 .. m-1 gebildet wird
  def multipliziere_mit_matrix(other_matrix)
  end
  # transponiert diese (nxm) Matrix und erzeugt eine
  # neue (mxn) Matrix m2.
  # Es gilt, dass m2[j][i] == diese_matrix[i][j]
  def transponiere_matrix()
  end
  # transponiert eine (nxn) Matrix, indem es die Inhalte
  # der Matrix tauscht
  def transponiere_matrix!()
  end
  # Ausgabe auf der Konsole
  def formatiert_ausgeben()
  end
end
```