Aufgabe 1: Zeigen Sie, dass die Untergruppe der Permutationsmatrizen in $\mathrm{GL}(n, \mathbb{R})$ isomorph zur symmetrischen Gruppe S_n ist.

Es sei Perm_n die Menge der Permutationsmatrizen in $\mathrm{GL}(n, \mathbb{R})$. Der Isomorphismus ist die Abbildung $\varphi : S_n \to Perm_n$ mit $\varphi(\sigma) \mapsto \sum_{i=1}^{n} e_{\sigma(i),i}$.

Dabei bezeichne $e_{j,i}$ die $n \times n$ Matrix, die eine Eins an der Stelle j, i und ansonsten nur Nullen hat. Anders ausgedrückt: die i-te Spalte von $\varphi(\sigma)$ ist der kanonische (Spalten-) Basisvektor $e_{\sigma(i)}$.

Die Abbildung ist umkehrbar, an den Spalten einer Permutationsmatrix liest man die Permutation σ ab. Außerdem ist die Umkehrung eindeutig. φ ist eine Bijektion.

Wir müssen noch zeigen, dass φ die Verknüpfung auf beiden Seiten respektiert, dass also für alle $\sigma_a, \sigma_b \in S_n$ gilt:

$$\varphi(\sigma_a) \cdot \varphi(\sigma_b) = \varphi(\sigma_a \circ \sigma_b) \tag{1}$$

Es seien $A := \varphi(\sigma_a)$ und $B := \varphi(\sigma_b)$ die Permutationmatrizen, die zu den Permutationen σ_a und σ_b gehören.

Die Wirkung von A auf einen Vektor $\vec{x}$ ist $A\vec{x}$. Die Zeilen von $\vec{x}$ werden durch σ_a permutiert. Ebenso ist $B\vec{x}$ die Permutation von $\vec{x}$ im Sinne von σ_b.

Nun ist aber $(AB)\vec{x} = A(B\vec{x})$. Rechts wird $\vec{x}$ zuerst durch σ_b und das Ergebnis anschließend durch σ_a permutiert. Damit ist AB die Permutationsmatrix, die zu $\sigma_a \circ \sigma_b$ gehört. Deswegen ist AB das Bild von $\sigma_a \circ \sigma_b$ unter φ. Wir können Gleichung (1) umformulieren als:

$$\begin{aligned} \varphi(\sigma_a) \cdot \varphi(\sigma_b) &= \varphi(\sigma_a \circ \sigma_b) \\ A \cdot B \quad &= \quad (AB) \end{aligned}$$

Aufgabe 2: Bestimmen Sie alle Automorphismen der Kleinschen Vierergruppe.

Ein Automorphismus ist bijektiv, deswegen ist ein Automorphismus eine **Permutation** auf der vierelementigen Gruppe.

S_4 hat $4! = 24$ Elemente. Da ein Automorphismus (als Homomorphismus) das neutrale Element fixiert, werden nur die übrigen drei Elemente permutiert, dafür gibt es $3! = 6$ Möglichkeiten.

Wie ein Blick in die Verknüpfungstabelle der Gruppe (z.B. in SAGE: `K=KleinFourGroup(); K.multiplication_table()`) zeigt, haben die drei übrigen Elemente jeweils die Ordnung 2, wir können sie nach Belieben permutieren.

Die Automorphismen von $K = \{x_0, x_1, x_2, x_3\}$, mit dem neutralen Element x_0, sind:

1. die Identität $\sigma_0(x) = x$ für alle $x \in K$
2. $\sigma_1 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_0 & x_2 & x_3 & x_1 \end{pmatrix}$
3. $\sigma_2 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_0 & x_3 & x_1 & x_2 \end{pmatrix}$
4. $\sigma_3 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_0 & x_1 & x_3 & x_2 \end{pmatrix}$
5. $\sigma_4 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_0 & x_3 & x_2 & x_1 \end{pmatrix}$
6. $\sigma_5 = \begin{pmatrix} x_0 & x_1 & x_2 & x_3 \\ x_0 & x_2 & x_1 & x_3 \end{pmatrix}$

Aufgabe 3: Bestimmen sie alle Isomorphieklassen von Gruppen der Ordnung 4.

Wir machen eine Fallunterscheidung: **Im ersten Fall** habe die Gruppe $G = \{0, a, b, c\}$ habe (mindestens) ein Element der Ordnung 4. O.B.d.A sei a dieses Element, es gilt also $< a >= \{a, 2a, 3a, 4a = 0\} = G$.

$2a$ hat Ordnung 2, wegen $2a + 2a = 4a = 0$, und $3a$ hat Ordnung 4, wegen $3a, 3a+3a = 6a = 4a+2a = 0+2a = 2a, 2a+3a = 5a = 4a+1a = 0+a, a+3a = 4a = 0$. Die Gruppe ist isomorph zu $\mathbb{Z}/4\mathbb{Z}$, siehe Abbildung 1.

+	0=4a	a	b=2a	c=3a
0	0	a	b	c
a	a	b	c	0
b	b	c	0	a
c	c	0	a	b

Abbildung 1: Die Verknüpfungstabelle im ersten Fall: es gibt ein Element der Ordnung 4.

Im zweiten Fall gebe es **kein** Element der Ordnung 4. In der Vorlesung (Satz von Lagrange) werdet Ihr noch kennenlernen, dass die Elementordnung die Gruppenordnung teilt. Es kommen also nur noch 1 und 2 als Ordnungen in Frage. (Die Annahme, dass es ein Element a mit der Ordnung 3 gibt, führt aber zu einem schönen Widerspruch, sobald man die Verknüpfungstabelle aufstellt.)

Wir nehmen also an, es gebe in G kein Element a der Ordnung 4. Sei $a \neq 0$ ein Element der Ordnung 2. ($\operatorname{ord} a = 1$ führt zu $a = 0$, ist also uninteressant und $\operatorname{ord} a = 3$ führt zu Widersprüchen.) Dann ist $a + a = 0$. Da G vier Elemente hat, gibt es neben 1 und a noch zwei weitere Elemente b, c die ebenfalls Ordnung 2 haben. (Ordnung 4 hatten wir ausgeschlossen, Ordnung 3 geht nicht und Ordnung 1 würde zu dem neutralen Element 0 führen). Siehe Abbildung 2 für die Verknüpfungstabelle.

Aufgabe 4: Implementieren Sie eine Funktion `def composition( f, g):` in SAGE, die für zwei Funktionen $f, g : X \to X$ ihr Kompositum zurückgibt. Hierbei ist X eine endliche Menge `( f.keys(), g.keys() )` und f und g sind als Dictionaries implementiert.

+	0	a	b	c
0	0	a	b	c
a	a	0	$\underline{c}$	$\underline{b}$
b	b	$\underline{c}$	0	$\underline{a}$
c	c	$\underline{b}$	$\underline{a}$	0

Abbildung 2: Die Verknüpfungstabelle im zweiten Fall: es gibt kein Element der Ordnung 4. Die unbestimmten Felder der Tabellen (hier unterstrichen) können wir so ausfüllen: pro Zeile und Spalte darf keine Element doppelt vorkommen.

Aufgabe 5: Bestimmen Sie mittels SAGE und der Funktion `composition( f, g)` aus Aufgabe 4 für alle Abbildungen $f : \{a, b\} \to \{a, b\}$ die Menge der Potenzen f^n (für $n = 0, 1, 2, \dots$) von f.

```
# Aufgabe 4:

# Bsp: X= 1,..,8
f = {1: 2, 2: 3, 3: 1, 4: 5, 5: 6, 6: 4, 7: 8, 8: 7}
g = {1: 1, 2: 5, 3: 6, 4: 3, 5: 4, 6: 8, 7: 2, 8: 7}

def composition( f, g ):
    """Gibt $f \\circ g$ zurück.

    f, g -- Dictionaries mit derselben Menge von Schlüsselwerten

    EXAMPLES:
        sage: f = {1: 2, 2: 3, 3:1}
        sage: g = {1: 3, 2: 1, 3:2}
        sage: composition(f,g)
        {1: 1, 2: 2, 3: 3}"""

    h= {} # leeres dictionary
    try:
        for k in g.keys():
            h[k] = f[ g[k] ] # f( g( k ) )
    except KeyError, e:
        print """Fehler: die Menge der Werte von g passt nicht zu
        der Menge der Schlüssel von f."""
    return h



print "Die Komposition von \n  ",f,"\nmit\n  ",g,\
      "\nist:\n  ",  composition(f , g),"\n\n"

# Die Komposition von
#    {1: 2, 2: 3, 3: 1, 4: 5, 5: 6, 6: 4, 7: 8, 8: 7}
# mit
#    {1: 1, 2: 5, 3: 6, 4: 3, 5: 4, 6: 8, 7: 2, 8: 7}
```

```
# ist:
#    {1: 2, 2: 6, 3: 4, 4: 1, 5: 5, 6: 7, 7: 3, 8: 8}

#-----------------------------------------------------------------

# Aufgabe 5:
def functionPower(f , n):
    """Berechnet f^n.

    Dabei ist f ein dictionary und n eine nicht negative ganze Zahl.

    EXAMPLES:
        sage: f = {'a': 'b', 'b':'a' }
        sage: functionPower(f, 0)
        {'a': 'a', 'b': 'b'}
        sage: functionPower(f, 1)
        {'a': 'b', 'b': 'a'}
        sage: functionPower(f, 2)
        {'a': 'a', 'b': 'b'}
    """

    tmp= {}

    if n < 0:
        print "Fehler, keine negativen Potenzen erlaubt."
        return None

    if n==0: # die Identität
        for k in f.keys():
            tmp[k] = k
    else:
        tmp = f
        for i in range(n-1):
            tmp = composition(tmp, f)
    return tmp


f1 = {'a': 'a', 'b':'b' }
f3 = {'a': 'a', 'b':'a' }
f2 = {'a': 'b', 'b':'a' }
f4 = {'a': 'b', 'b':'b' }

for n in range(5):
    print "f4^"+str(n)+":", functionPower(f4,n)

# f1:
# f1^0: {'a': 'a', 'b': 'b'}
# f1^1: {'a': 'a', 'b': 'b'}
# f1^2: {'a': 'a', 'b': 'b'}
# f1^3: {'a': 'a', 'b': 'b'}
# f1^4: {'a': 'a', 'b': 'b'}
```

```
# f2:
# f2^0: {'a': 'a', 'b': 'b'}
# f2^1: {'a': 'b', 'b': 'a'}
# f2^2: {'a': 'a', 'b': 'b'}
# f2^3: {'a': 'b', 'b': 'a'}
# f2^4: {'a': 'a', 'b': 'b'}

# f3:
# f3^0: {'a': 'a', 'b': 'b'}
# f3^1: {'a': 'a', 'b': 'a'}
# f3^2: {'a': 'a', 'b': 'a'}
# f3^3: {'a': 'a', 'b': 'a'}
# f3^4: {'a': 'a', 'b': 'a'}


# f4:
# f4^0: {'a': 'a', 'b': 'b'}
# f4^1: {'a': 'b', 'b': 'b'}
# f4^2: {'a': 'b', 'b': 'b'}
# f4^3: {'a': 'b', 'b': 'b'}
# f4^4: {'a': 'b', 'b': 'b'}
```