Prof. Dr. Marcin Grzegorzek
Research Group for Pattern Recognition
Lehrstuhl für Mustererkennung
www.pr.informatik.uni-siegen.de

Einführung in die Informatik I
Übung 4

1 Matrizen: Überlagerung der Vektoren

Schreiben Sie eine MATLAB-Funktion namens **`Quersumme`**, die als Parameter eine quadratische (n x n) Matrix M erhält. Die Funktion soll prüfen, ob die Summe einer Zeile mit der jeweiligen Summe der Spalte übereinstimmt. Bei Übereinstimmung wird in einen Vektor `y` eine `1` eingetragen, andernfalls eine `0`.

Bsp.: 3x3-Matrix

$$M = \begin{bmatrix} 8 & 3 & 6 \\ 4 & 8 & 7 \\ 4 & 9 & 5 \end{bmatrix} \longrightarrow \text{Ergebnis: } \texttt{y = [0 0 1]}$$

$$\sum Spalte1 \neq \sum Zeile1 \rightarrow y(1) = 0$$
$$\sum Spalte2 \neq \sum Zeile2 \rightarrow y(2) = 0$$
$$\sum Spalte3 = \sum Zeile3 \rightarrow y(3) = 1$$

Hinweis: Verwenden Sie den **`sum`**-Befehl.

2 Matrix-Zugriff per Colon(:)-Operator

Erzeugen Sie eine 5x5-Matrix $A = (a_{ij})_{5x5}$ durch die folgende Vorschrift für Elemente

$$a_{ij} = j + 5(i - 1)$$

Diskutieren Sie die Ergebnisse der folgenden Zugriffsoperationen:

a) `A(2, :), A(:, 5)` und `A(:,:)`
b) `A(1, 1 : 2 : 5)`
c) `A(3 : -1 : 1 , 5 : -1 : 3)`

3 Fehlerfinden

Suchen Sie die Stellen in folgenden Funktionen, die eine Fehler- oder Warnmeldung verursachen. Bestimmen Sie die **Zeile(n) mit einem Fehler** und **begründen** Sie Ihre Entscheidung. Es ist auch möglich, dass eine oder mehrere Funktionen **fehlerfrei** sein können. Schreiben Sie als Begründung FEHLERFREI, wenn das Skript vollständig funktioniert.

Prof. Dr. Marcin Grzegorzek
Research Group for Pattern Recognition
Lehrstuhl für Mustererkennung
www.pr.informatik.uni-siegen.de

UNIVERSITÄT SIEGEN

```
function v = f1(x,n)
v = zeros(n,1);
for i = 1:round(n/2)+3x
v(i) = n * x;
end
```
1

```
function x = f2(b)
if (cos(b)^2-(sin(b+1)))*b)> 0
x = 1
else
x = 0
end
```
2

```
function acht = f3(neun, eins)
v = [eins;3.14];
zwei=round(v(2));
if pi > v(zwei)
drei = 1 + zwei;
else
fuenf = 2 + zwei;
end
acht = drei + fuenf;
```
3

```
function y = f4(b)
y = 1;
while x + y < 10
x = y + 1;
y = x + y;
end
```
4

```
function beta=f5(alpha)
if length(pi) = length(2*pi)
beta = -alpha;
elseif length(pi) < length(2*pi)
beta = +alpha;
else
beta = alpha+alpha/2;
end
end
```
5

```
function x=f6(u)
if length(u+f6) > 3
x=u+log10(u);
else
x=u-log10(u);
end
end
```
6

```
function x=f7(t)
x=zeros(length(t),length(t));
for i=1:size(x,1)
for j=i:length(t)+1
x(i,j)=t(j);
end
end
end
```
7

```
function x=f8(y)
f=y+3;
x=sqrt((2+y)/(cos(3*f)+y))+1;
x=x+1;
end
```
8

```
function x=f9(y)
h=y+3;
x=sin((2+y/ln(h))/(tan(h*z)+pi));
x=x+1;
end
```
9

```
function x=f10(s)
if length(s) > 3
x=1;
else
x=0;
end
end
```
10

```
function x=f11(w)
for i=1:length(w)
w(i)==w(i)+log(w(i));
end
x=w;
```
11

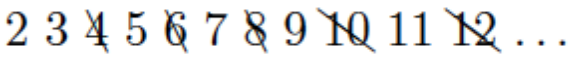
```
function x=f12(y)
z=y+factorial(5);
x=diff((log(y))/((pi*z)+y)+1);
x=x+1;
end
```
12

Prof. Dr. Marcin Grzegorzek
Research Group for Pattern Recognition
Lehrstuhl für Mustererkennung
www.pr.informatik.uni-siegen.de

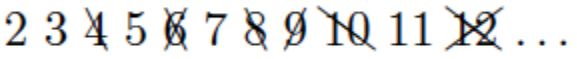
UNIVERSITÄT SIEGEN

3 Sieb des Eratosthenes

Das Sieb des Eratosthenes ist eine einfache Methode zur Bestimmung aller Primzahlen p im Intervall $2 \leq p \leq n$. Dazu werden die Zahlen 2 bis n in eine Reihe geschrieben. Sodann wird die erste nicht durchgestrichene Zahl p (zu Beginn also 2) genommen und jedes Vielfache von p, außer p selbst, durchgestrichen:



Nun wird die nächste nicht durchgestrichene Zahl gesucht (also 3) und deren Vielfache durchgestrichen:



Sobald die Vielfachen aller Zahlen der Liste durchgestrichen wurden, sind alle Primzahlen bis n gefunden.

Aufgabenstellung: Programmieren Sie das Sieb des Eratosthenes als Funktion namens **`Eratosthenes`**, die das Algorithmus durchführt.