# Funktionen und Kontrollstrukturen

## Übungsaufgabe 4*

Eine **Primzahltabelle** bis n ist eine Liste aller Primzahlen zwischen 1 und n. Für $n = 10$ lautet diese: $2, 3, 5, 7$. Für jede Zahl i, die keine Primzahl ist, gibt es zwei Zahlen j und k, $2 \leq j, k \leq n$, für die gilt, dass $i = j \cdot k$. Damit kann folgender Algorithmus zur Erstellung einer Primzahltabelle abgeleitet werden:

1. Definiere einen Vektor `prims` mit den ganzen Zahlen 2 bis n.
2. Für alle $j = 2, \ldots, n$ und für alle $k = 2, \ldots, n$, berechne $i = j \cdot k$.
3. Entferne jeweils das Ergebnis $i$, falls in `prims` vorhanden, aus dem Vektor `prims`.

Alle Zahlen, die am Ende im Vektor `prims` vorkommen, sind Primzahlen.

a) Schreiben Sie eine Funktion `prime_table1`, die eine Primzahltabelle bis zu einer Zahl n nach obiger Vorschrift berechnet und zurückgibt. Die Grundstruktur der Funktion soll folgendermaßen aussehen:

```
prime_table1 <- function(n){
...
...
...
}
```

Eine deutlich effizientere Version des Verfahrens ist das sogenannte **Sieb des Eratosthenes**, welches folgendermaßen definiert ist:

1. Definiere einen Vektor `prims` mit den ganzen Zahlen 2 bis n.
2. Für alle $j = 2, \ldots, \lfloor \sqrt{n} \rfloor$, falls in `prims` vorhanden, und für alle $k = j, \ldots, \lfloor \frac{n}{j} \rfloor$, berechne $i = j \cdot k$.
3. Falls $n \neq 2$, entferne jeweils das Ergebnis $i$, falls in `prims` vorhanden, aus dem Vektor `prims`.

Alle Zahlen, die am Ende im Vektor `prims` vorkommen, sind Primzahlen.

b) Schreiben Sie eine Funktion `prime_table2`, die eine Primzahltabelle bis zu einer Zahl n mithilfe des Sieb des Eratosthenes berechnet und zurückgibt. Die Grundstruktur der Funktion soll folgendermaßen aussehen:

```
prime_table2 <- function(n){
...
...
...
}
```

c) Berechnen Sie mithilfe der Funktionen `prime_table1` und `prime_table2` die Primzahltabellen für die Zahlen $n = 50$ und $n = 150$. Überprüfen Sie, ob beide Funktionen jeweils zum selben Ergebnis führen.

d) Schreiben Sie eine Funktion `check.input`, die überprüft ob

- die Eingabe $\geq 2$ ist. Falls nicht, soll die Funktion das Programm abbrechen und eine entsprechende Fehlermeldung ausgeben.
- ob der Input eine natürliche Zahl ist. Falls nicht, soll die Funktion den Input runden und das Programm weiter ausführen. Weiterhin soll eine entsprechende Warnmeldung ausgegeben werden.

Die Grundstruktur der Funktion soll folgendermaßen aussehen:

```
check.input <- function(n){
...
...
...
}
```

e) Fügen Sie die Funktion `check.input` an geeigneter Stelle in die Funktionen `prime_table1` und `prime_table2` ein.

f) Testen Sie nun die Funktionalität der Funktionen `prime_table1` und `prime_table2`, indem Sie folgende Werte für n als Input übergeben: -1 0 1.5 2 10 10.2 10.6

** Durch Lösen dieser Übungsaufgabe kann ein Teil der Prüfungsleistung erbracht werden.*

**Abgabetermin: 20.06.2017/21.06.2017**