Aufgabe 6.1 Mit Zufallszahlen vertraut werden

Das folgende Beispiel zeigt die idiomatische Erzeugung von Pseudo-Zufallszahlen in C:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int i;
    srand(time(0));   /* Generator EINMALIG initialisieren */

    for (i = 0; i < 10; ++i)
    {
        printf("Der Wuerfel zeigt eine %d.\n", 1 + rand() % 6);
    }
}
```

Tippe das Programm ab und probiere es aus. Welche möglichen Werte liefert `rand()`? Welche möglichen Werte liefert `rand()%6`? Welche möglichen Werte liefert `1+rand()%6`?

Ein beliebter Anfängerfehler ist es, `srand` mehr als einmal aufzurufen, zum Beispiel innerhalb der Schleife unmittelbar vor dem Aufruf von `rand`. Was passiert dann?

Was passiert, wenn man `srand` überhaupt nicht aufruft?

Was passiert, wenn man `srand` mit einem anderen Argument aufruft, z.B. `42`?

Mit einem Blick auf die Implementation sollte das Verhalten besser nachvollziehbar sein:

```
unsigned state = 1;

void srand(unsigned seed)
{
    state = seed;
}

int rand()
{
    state = state * 214013 + 2531011;
    return state << 1 >> 17;
}
```

Wie Du siehst, verläuft die Berechnung von Zufallszahlen komplett deterministisch.

Die Funktion `srand` initialisiert den Zufallsgenerator auf den übergebenen Startwert und determiniert damit, welche Zufallszahlen anschließend von `rand` geliefert werden.

Die Initialisierung mittels `srand` sollte **einmalig** innerhalb von `main` geschehen! **Auf keinen Fall solltest du `srand` und `rand` abwechselnd aufrufen.** Das macht die Zufallszahlen nicht zufälliger, wovon Du Dich oben bereits überzeugen konntest.

Aufgabe 6.2 Zahlenraten

Programmiere ein Spiel, bei dem der Computer sich eine Zahl zwischen 1 und 100 ausdenkt und der Spieler die Zahl erraten soll. Der Computer muss nach jedem Rateversuch wahrheitsgemäß antworten, ob seine Zahl kleiner, größer oder gleich ist.

Die Anzahl der Rateversuche soll protokolliert werden. Wenn die Zahl erraten wurde, darf der Spieler auf Wunsch noch einmal spielen.

```
Ich habe mir eine Zahl zwischen 1 und 100 ausgedacht. Ihre
Aufgabe ist es, diese Zahl zu erraten!
? 20
Meine Zahl ist größer!
? 60
Meine Zahl ist kleiner!
? 55
Meine Zahl ist größer!
? 57
Korrekt! Sie haben 4 Versuche gebraucht.
Wollen Sie noch einmal spielen? n
```

Aufgabe 6.3 Highscore

Erstelle per Hand (Notepad o.ä.) eine Textdatei `highscore.txt` mit folgendem Inhalt:

```
10 Max
```

Lies den Highscore am Anfang von *Zahlenraten* ein. Sobald der aktuelle Spieler das Spiel mit weniger Versuchen beendet, dann soll eine Meldung folgender Art erscheinen:

```
Glückwunsch, Sie haben den Highscore von Max (10) geschlagen!
Wie lautet Ihr Name?
```

Dieser neue Highscore soll wiederum in `highscore.txt` abgespeichert werden.

Aufgabe 6.4 Top 10

Erweitere *Zahlenraten* dergestalt, dass nicht nur der beste Spieler gespeichert wird, sondern die 10 besten. Diese sollen am Anfang des Spiels auf die Konsole geschrieben sowie immer dann, wenn der aktuelle Spieler einen Platz in der Top 10 ergattert.

Aufgabe 6.5 Sieb des Eratosthenes

Schreibe eine Funktion `int * berechne_primzahlen(int grenze)`, die alle Primzahlen kleiner als `grenze` berechnet und zurückgibt. Dazu eignet sich ein effizienter Algorithmus, der als „Sieb des Eratosthenes“ (siehe Wikipedia) bekannt geworden ist:

- Erzeuge ein Array `prim` mit `grenze` Zellen, die (mit Ausnahme der ersten beiden Zellen) alle auf 1 initialisiert sind.
- Durchlaufe `i` ab `2` für alle Zahlen kleiner als `grenze`.
- Wenn `prim[i]` gleich 1 ist, ist `i` eine Primzahl. Alle Vielfachen von i (2*i, 3*i, 4*i, …) sind keine Primzahlen und müssen daher im Array mit 0 markiert werden.

Wenn der Algorithmus durchgelaufen ist, sind alle Primzahlen in `prim` mit 1 markiert. Kopiere diese Primzahlen in ein neues, passendes Array und liefere dieses anschließend zurück. (Schließlich möchte der Aufrufer keine Liste von Nullen und Einsen erhalten, sondern eine Liste mit den Zahlen 2, 3, 5, 7...)

Denke Dir eine Konvention aus, anhand welcher der Aufrufer aus dem Array entweder das Ende oder die Größe des Arrays auslesen kann. Dokumentiere diese Konvention.

Aufgabe 6.6 Zahlenraten mit vertauschten Rollen

Programmiere ein Spiel, bei dem der Spieler sich eine Zahl zwischen 1 und 100 ausdenkt und der Computer diese Zahl erraten muss. Dazu fragt er den Spieler, ob es sich um eine Zahl x handelt. Der Spieler muss wahrheitsgemäß antworten, ob x korrekt ist oder zu klein bzw. zu groß gewählt wurde, woraufhin der Computer das nächste x sinnvoll einschränkt.

```
Bitte denken Sie sich eine Zahl aus. Antworten Sie im Folgenden
immer wahrheitsgemäß, ob ihre Zahl kleiner (-), größer (+) oder
gleich (=) ist!
Ist es die 50? -
Ist es die 25? -
Ist es die 12? -
Ist es die 6? +
Ist es die 9? =
Sie haben an die Zahl 9 gedacht!
Dafür habe ich 5 Versuche gebraucht.
```