Aufgabe 7.1 Fakultät (Pflicht)

Die Fakultätsfunktion ist mathematisch wie folgt definiert:

```
fak(0) = 1
fak(n) = n * fak(n - 1)
```

In C kann diese Definition 1:1 rekursiv umgesetzt werden:

```
int fak(int n)
{
    if (n == 0)
        return 1;
    else
        return n * fak(n - 1);
}
```

Schreibe eine iterative Variante von `fak`, also mit einer Schleife anstatt mit Rekursion.

Aufgabe 7.2 Fibonacci (Pflicht)

Die Fibonaccifunktion ist mathematisch wie folgt definiert:

```
fib(0) = 0
fib(1) = 1
fib(n) = fib(n - 2) + fib(n - 1)
```

Setzte diese Definition 1:1 rekursiv in C um, ohne Schleifen oder Arrays.

Aufgabe 7.3 Mitzählen (Pflicht)

Für große `n` explodiert der Berechnungsaufwand, weil sich die Funktion `fib` jedes Mal doppelt rekursiv aufruft. Zähle mit, wie oft die Funktion `fib` insgesamt aufgerufen wird. Definiere zu diesem Zweck eine entsprechende Zählvariable oberhalb von `fib`.

Wie oft wird `fib` aufgerufen, um `fib(10)`, `fib(20)` oder `fib(30)` zu berechnen?

Aufgabe 7.4 Binär (Pflicht)

Die rekursive Funktion `print_binary` schreibt eine Zahl in binär auf die Konsole:

```
void print_binary(unsigned x)
{
    if (x >= 2)
        print_binary(x / 2);
    putchar("01"[x % 2]);
}
```

Erkläre deinem Betreuer anhand des Aufrufs `print_binary(42)`, wie diese Funktion arbeitet. Welche Werte nehmen die Parameter `x` nacheinander an?