Wählen Sie aus den folgenden Aufgaben zur Bearbeitung drei Aufgaben aus.

2 C-Syntax I

Aufgabe 2.1. (Schaltjahrberechnung – leicht) Ein Schaltjahr ist ein Jahr, welches durch 400 teilbar ist, oder zwar durch 4 aber nicht durch 100 teilbar ist. Dies entspricht der Formel

$$L(Y) = 4 \mid Y \wedge 100 \nmid Y \vee 400 \mid Y.$$

Schreiben Sie ein Programm, dass sein erste Argument ausliest und berechnet ob es sich um ein Schaltjahr handelt. Kommt Ihr Programm mit der falschen Anzahl an Argumenten zurecht? Wie ist es mit Eingaben, die keine Zahlen sind?

Codebeispiel 1: Beispielausgabe

```
$ ./leap 2008
The year 2008 is a leap year.
$ ./leap 2100
The year 2100 is not a leap year.
$ ./leap 2000
The year 2000 is a leap year.
```

Aufgabe 2.2. (Ostertagberechnung – leicht) Implementieren Sie den Algorithmus von Gauß zur Bestimmung des Ostertages [1] in einem gegebenen Jahr. Nehmen Sie die Jahreszahl als Argument entgegen.

Codebeispiel 2: Beispielausgabe

```
$ ./easter 2024
The easter date of the year 2024 is the 31. of March
$ ./easter 2018
The easter date of the year 2018 is the 1. of April
```

Aufgabe 2.3. (Funktionstafel – leicht) Schreiben Sie ein Programm, dass für die Zahlen $x = 1, 2, \ldots, 100$ die Werte der Ausdrücke x^2, 2^x, und $x!$ bestimmt und tabellarisch ausgibt. Versuchen Sie, die Ausgabe ansprechend zu gestalten, in dem Sie die Zahlen rechts ausgerichtet übereinander schreiben. Lesen Sie ggf. **printf**(3) um zu verstehen, wie man solche Ausrichtung durchführen kann

Vergleichen Sie die Ausgabe Ihres Programmes mit den erwarteten Werten. Sind sie identisch? Falls nicht, warum?

Aufgabe 2.4. (wc(1) – mittel) Schreiben Sie einen einfachen Klon des klassischen UNIX-Programms **wc**(1). Das Programm **wc**(1) ließt die Eingabe und zählt, wie viele Zeichen, Wörter, und Zeilen in ihr vorkommen. Denken Sie darüber nach, wie Sie Wortgrenzen und Zeilenenden erkennen können. Geben Sie am Ende die Anzahlen mit **printf**(3) aus.

Rufen Sie **getchar**(3) in einer Schleife wie folgt auf, um die Eingabe zu lesen. Sie können Ihr Programm dann folgendermaßen aufrufen:

```
./mein_wc <datei.txt
```

```c
    int c;

    /* lese ein Zeichen in c, und solange c nicht EOF
     * (End Of File, Ende der Eingabe) ist, ... */
    while (c = getchar(), c != EOF) {
        /* tue irgendwas mit c */
    }

    /* ist die Schleife beendet, wurde das Eingabeende erreicht */
```

Aufgabe 2.5. (rot13 – mittel) Schreiben Sie ein Programm, dass von der Eingabe Text ließt und diesen nach der „ROT13“-Methode verschlüsselt. Dazu wird jeder Buchstabe durch den Buchstaben 13 Stellen weiter (oder vorher) im Alphabet ersetzt. Benutzen Sie das gleiche Skelett wie bei der **wc**-Aufgabe. Sie können das eingelesene Zeichen nach der Umwandlung mit Hilfe der Funktion **putchar**(3) wieder ausgeben.

Aufgabe 2.6. (Bitfunktionen – leicht) Schreiben Sie C-Funktionen zur Berechnung der folgenden Bit-Operationen. Jede Funktion soll ein Argument vom Typ `unsigned int` entgegen nehmen und das Ergebnis erneut als `unsigned int` ausgeben.

sadd(x) zählt, wie viele Bits in `x` gesetzt sind

ctz(x) zählt, wie oft `x` glatt durch 2 teilbar ist

rev(x) kehrt die Reihenfolge der Bits von `x` um

bswap(x) kehrt die Reihenfolge der Bytes von `x` um

Schreiben Sie ein Programm, das Ihre Implementierungen ausprobiert und vergewissern Sie sich ihrer Korrektheit.

Versuchen Sie, Ihre Implementierung hinsichtlich der Performanz zu optimieren.

Aufgabe 2.7. (Datum aus Systemzeit – schwer) Die Funktion **time**(2) (je nach System ggf. **time**(3)) liefert die aktuelle Zeit in Sekunden seit dem 1. Januar 1970, 00:00 UTC zurück.

Schreiben Sie ein Programm, das aus der Rückgabe von **time**(2) Zeit, Datum, Wochentag, und Uhrzeit bestimmt. Alternativ soll es dem Nutzer möglich sein, eine zu konvertierende Sekundenzahl per Argument an das Programm zu übergeben.

Denken Sie sich selbst einen Algorithmus aus oder recherchieren Sie im Internet. Sie können die Ausgabe Ihres Codes anschließend mit dem von **ctime**(3) bestimmen String vergleichen. Beachten Sie dabei, dass der Ausgabe von **ctime**(3) abhängig von der aktuellen Zeitzone ist.