1. MinMax

Lösungsidee:

Das Programm instanziiert zwei Variablen min und max mit 0. Wird mehr als ein Parameter übergeben, werden die Parameter in einer FOR Schleife durchgegangen. Der Wert wird in jeder Iteration mit der Standardfunktion atoi() aus dem Array ausgelesen. Dadurch wird das Programm auch bei der Eingabe von Text richtig ausgeführt, da die Funktion im Falle eines Strings einfach 0 zurückgibt. Ist der Wert unter 0 und kleiner als der aktuelle Wert von min, wird der Wert in min gespeichert. Ist der Wert über null und größer als der aktuelle Wert von max, wird der Wert in max gespeichert. Am Ende werden die beiden Werte ausgegeben.

Code:

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

int main (int argc, char *argv[]) {
    int min = 0, max = 0;
    int value;

    if (argc <= 1) {
        printf("Please add more numbers\n");
    } else {
        for (int i = 1; i < argc; i++) {
            value = atoi(argv[i]);

            if (value >= 0) {
                if (value > max)
                    max = value;
            } else {
                if (value < min)
                    min = value;
            }
        }
        printf("Minimum: %d | Maximum: %d\n", min, max);
    }

    return 0;
}
```

Testfälle:

```
root@a3ea2cf6de4c:/home/swo3/src/minmax# ./minmax 1 -1 2 -2 3 -3 4 -4
Minimum: -4 | Maximum: 4
```

Einfacher Test mit Zahlen von -4 bis 4

```
root@a3ea2cf6de4c:/home/swo3/src/minmax# ./minmax 10 0 59 9 ad 9 0 9 -23 -32 -400 lk 200
Minimum: -400 | Maximum: 200
```

Test verschiedener Zahlen gemischt mit Strings

```
root@a3ea2cf6de4c:/home/swo3/src/minmax# ./minmax
Please add more numbers
```

Test ohne Parameter. Hier wird eine Warnung ausgegeben.

```
root@a3ea2cf6de4c:/home/swo3/src/minmax# ./minmax 1
Minimum: 0 | Maximum: 1
```

Eingabe eines einzelnen Parameters. Wie gefordert bleibt min auf 0, da keine negative Zahl eingegeben wurde.

```
root@a3ea2cf6de4c:/home/swo3/src/minmax# ./minmax -123
Minimum: -123 | Maximum: 0
```

Wie der vorherige Test, nur mit einer negativen Zahl.

2. Primfaktorenzerlegung

Lösungsidee:

Das Programm geht in einer FOR Schleife alle eingegeben Parameter durch. Der Wert wird dem Eingabearray mit der Standardfunktion atoi() entnommen und in der Variable VALUE gespeichert. Wird ein String eingegeben, liefert diese Funktion den Wert 0. Da eine Unterscheidung zwischen der Eingabe eines Strings und des Werts 0 den Rahmen der Übung sprengen würde, wurde darauf verzichtet.
Ist der Wert unter 0 wird eine entsprechende Fehlermeldung ausgegeben und das Programm beendet. Sonst wird die Zahl + { ausgegeben. Eine weitere FOR Schleife zählt von 2 bis VALUE hoch. Mithilfe einer Hilfsfunktion wird überprüft, ob die aktuelle Zählvariable eine Primzahl ist. Ist das der Fall, wird VALUE so lange mit der primzahl dividiert, bis bei der Division ein Rest entsteht. Dabei wird mitgezählt, wie oft die Division durchgeführt wurde. Ist der Counter > 0 wird die Zählvariable und die Anzahl der Divisionen ausgegeben. Hat die Zählvariable den Wert von VALUE erreicht oder überschritten und VALUE ist eine Primzahl, so wird VALUE mit der Anzahl 1 ausgegeben. Dadurch werden auch die Sonderfälle von 1 und 0 berücksichtigt.

isPrime(int x): In einer FOR Schleife wird von 2 bis x / 2 gezählt. In der Schleife wird geprüft, ob x % i == 0 ist. Wenn das einmal zutrifft, ist x keine Primzahl.

Code:

```c
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
    int value, count;
    for (int i = 1; i < argc; i++) {
        value = atoi(argv[i]);
        if (value < 0) {
            printf("Number has to be > 0\n");
            return 0;
        } else {
            printf("%10d: {", value);
            for (int prim = 2; prim < value; prim++) {
                if (isPrime(prim)) {
                    count = 0;
                    while (value % prim == 0) {
                        value /= prim;
                        count++;
                    }
                    if (count != 0)
                        printf("{%d,%d},", prim, count);
                }
            }
            if (isPrime(value))
                printf("{%d,1}", value);
            printf("}\n");
        }
    }
    return 0;
}
```

```
int isPrime(int x) {
    for (int i = 2; i <= x / 2; ++i) {
        if (x % i == 0)
            return 0;
    }

    return 1;
}
```

Testfälle:

```
root@b37df0263295:/home/swo3/src/prim# ./prim 35000 0 42 1 65537
    35000: {{2,3},{5,4},{7,1}}
        0: {{0,1}}
       42: {{2,1},{3,1},{7,1}}
        1: {{1,1}}
    65537: {{65537,1}}
```

Test mit den in der Angabe gezeigten Zahlen.

```
root@b37df0263295:/home/swo3/src/prim# ./prim 3 5 7 11 17 23 677
        3: {{3,1}}
        5: {{5,1}}
        7: {{7,1}}
       11: {{11,1}}
       17: {{17,1}}
       23: {{23,1}}
      677: {{677,1}}
```

Test mit nur Primzahlen

```
root@b37df0263295:/home/swo3/src/prim# ./prim 0 1
        0: {{0,1}}
        1: {{1,1}}
```

Test mit 0 und 1

```
root@b37df0263295:/home/swo3/src/prim# ./prim 6 10 65538
        6: {{2,1},{3,1}}
       10: {{2,1},{5,1}}
    65538: {{2,1},{3,2},{11,1},{331,1}}
```

Test mit nicht-Primzahlen.

```
root@b37df0263295:/home/swo3/src/prim# ./prim 214748364
 214748364: {{2,2},{3,1},{29,1},{43,1},{113,1},{127,1}}
```

Test mit einer sehr großen Zahl.

```
root@b37df0263295:/home/swo3/src/prim# ./prim
root@b37df0263295:/home/swo3/src/prim#
```

Test ohne Parameter