SWO3	**Übung zu Softwareentwicklung mit klassischen Sprachen und Bibliotheken 3**	**WS 2021/22, ÜZ 1**

☐ Gruppe M. Hava

☒ Gruppe J. Heinzelreiter Name: Angelos Angelis Aufwand [h]: 4

☐ Gruppe P. Kulczycki Peer-Review von: _____

Beispiel	Lösungsidee (max. 100%)	Implement. (max. 100%)	Testen (max. 100%)
1 (40 P)	100%	100%	100%
2 (60 P)	100%	90%	100%

Beispiel 1: MinMax (`src/minmax/`)

Schreiben Sie ein C11-Programm `minmax`, das das Folgende leistet: Dem Programm können auf der Kommandozeile beliebig viele negative und positive ganze Zahlen mitgegeben werden. Das Programm muss die kleinste negative (Minimum) sowie die größte positive (Maximum) der mitgebenen Zahlen als Ergebnis auf der Konsole ausgegeben. Kommen in der Parameterliste keine negativen Zahlen vor, so muss für das Minimum `0` ausgegeben werden. Analoges gilt für das Maximum.

Beispiel 2: Primfaktorenzerlegung (`src/prim/`)

Schreiben Sie ein C11-Programm factorinteger, welches die Primfaktoren aller positiven ganzen Zahlen, die aus der Kommandozeile gelesen werden, berechnet. Die Primfaktoren müssen dabei aufsteigend und mit ihrer Multiplizität auf der Konsole ausgegeben werden. Die Ausgabe muss in dem in den Beispielen angegebenen Format erfolgen:

```
./factorinteger.exe 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}}
```

Aufgabe 1)

Lösungsidee:

Man durchläuft mittels einer For-Schleife alle Argumente. Diese geht man dann der Reihe nach und überprüft ob jeweils jede Zahl Negativ oder Positiv ist und überprüft auch dementsprechend ob sie kleiner der bisher kleinsten gespeicherten Zahl oder größer als der bisher größten gespeicherten Zahl ist. Falls ja speichert man den Wert dementsprechend ein.

Code:

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

int main(int argc, char *argv[]){

   int min = 0;
   int max = 0;
   int currentNumber;

   for (int i = 1; i < argc; i++) /* Für jedes Argument */
   {
      currentNumber = atoi(argv[i]);
      if (currentNumber < 0 && min > currentNumber)
      {
         min = currentNumber;
      }
      if (currentNumber > 0 && max < currentNumber)
      {
         max = currentNumber;
      }
   }

   printf("min=%-d\nmax=%d\n",min,max);

   return EXIT_SUCCESS;

}
```

Testfälle:

1) 0 und mehrere positive Zahlen

```
root@e3b24ab556f4:/home/swo3# ./MinMax 0 10 20 40 10 5
min=0
max=40
```

2) 0 und mehrere negative Zahlen

```
root@e3b24ab556f4:/home/swo3# ./MinMax -10 0 -20 -15 -40 -35
min=-40
max=0
```

3) Positive und negative Zahlen

```
root@e3b24ab556f4:/home/swo3# ./MinMax -10 0 -20 -15 -40 -35 10 20 30 40 35
min=-40
max=40
```

Aufgabe 2)

Lösungsidee:

Man durchläuft mittels einer For-Schleife alle Argumente. Diese geht man dann der Reihe nach und macht Folgendes:

1) Man findet mittels einer While Schleife den nächsten Primfaktor
2) Teilt diesen durch unsere Zahl und speichert das Ergebnis und den Teiler(bzw Primfaktor)
3) Man gibt das Ergebnis im gewünschten Format aus

Zu beachten ist, das in meinem Programm die Zahl 0 und 1 extra behandelt wird.

Code:

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

int main(int argc, char *argv[]){

   int currentNumber;

   for (int i = 1; i < argc; i++) /*für jedes Argument*/
   {
      currentNumber = atoi(argv[i]);

      if (currentNumber < 0) /*negative Zahlen müssen extra behandelt werden */
      {
         printf("Only positive Numbers\n");
         return EXIT_FAILURE;
      }
      else if (currentNumber == 1) /* 1 muss extra behandelt werden */
      {
         printf("%d: {",1);
         printf("{%d,%d}}\n",1,1);

      }
      else if (currentNumber == 0)/* 0 muss extra behandelt werden */
      {
         printf("%d: {",0);
         printf("{%d,%d}}\n",0,1);

      }
      else {
         int j = 2;
         int primfaktor = 2;
         int exponent = 0;
```

```
            printf("%d: {",currentNumber);
            while (j < currentNumber)
            {
               while(currentNumber%j != 0){ /* Finde den nächsten Primfaktor */
                  j++;
               }
               while (currentNumber%j == 0){
                  currentNumber /=j;
                  if (primfaktor == j)
                  {
                     exponent++;
                  }else{
                     if (exponent != 0)
                     {
                     printf("{%d,%d},",primfaktor,exponent);
                     }
                     primfaktor = j;
                     exponent = 1;
                  }
               }
            }
            printf("{%d,%d}",primfaktor,exponent);
            printf("}");
            printf("\n");
         }
      }
      return EXIT_SUCCESS;
}
```

Testfall:

1) Die Zahlen der Angabe und eine Negative Zahl

```
root@e3b24ab556f4:/home/swo3# ./factorinteger 35000 0 42 1 65537 -2
35000: {{2,3},{5,4},{7,1}}
0: {{0,1}}
42: {{2,1},{3,1},{7,1}}
1: {{1,1}}
65537: {{65537,1}}
Only positive Numbers
```