Solutions to Lars's Übung 2

3. April 2017

1 Aufgabe 1

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */

#include <stdio.h>

int main () {
    int n = 10;       /* Addiere bis zu dieser Zahl */
    int i;
    int summe;        /* speichert Zwischenergebnis */

    i = 0;

    while (i < n) {
      summe =+ i      /* addiere ite Zahl auf summe */
    }
    printf ("Das Ergebnis ist %f.\n", summe);
    return 0;
}
```

First of all we have to check whether the actual implementations compile. Using `gcc` we get the following error messages:

```
temp2.c: In function 'main':
temp2.c:15:5: error: expected ';' before '}' token
     }
     ^
temp2.c:16:13: warning: format '%f' expects argument of type 'double', but argument
    2 has type 'int' [-Wformat=]
      printf ("Das Ergebnis ist %f.\n"
```

The first error message misses the end of the instruction at the 15-th line. The second message is only a warning. It warns us that we print with `%f` a variable of `int` typ. This should be `%d`. The source code, that can be successfully compiled look like the following:

```c
/* Summe der ersten n Zahlen.
 * (c) 2015 Clelia und Johannes */
```

```
3
4  #include <stdio.h>
5
6  int main () {
7      int n = 10;       /* Addiere bis zu dieser Zahl */
8      int i;
9      int summe;        /* speichert Zwischenergebnis */
10
11     i = 0;
12
13     while (i < n) {
14       summe =+ i;      /* addiere ite Zahl auf summe */
15     }
16     printf ("Das Ergebnis ist %d.\n", summe);
17     return 0;
18 }
```

However, even if the code compiles it does not guarantee a meaningfull output. Running the code, it seems it runs forever. This is because the statements in `while` actually never became falls, since `i` is not incremented inside `while`.This should be done with `i++;`.Besides this, we have to initialize all variables correctly. This can be done together with the declaration. Comitting the above changes our code will give meaningfull results.

```
1  /* Summe der ersten n Zahlen.
2   * (c) 2015 Clelia und Johannes */
3
4  #include <stdio.h>
5
6  int main () {
7      int n = 10;       /* Addiere bis zu dieser Zahl */
8      int i=0;          /* initializierung */
9      int summe=0;       /* speichert Zwischenergebnis */
10
11     while (i < n) {
12       summe =+ i;      /* addiere ite Zahl auf summe */
13       i++;
14     }
15     printf ("Das Ergebnis ist %d.\n", summe);
16     return 0;
17 }
```

Although our code now gives meaningfull results, this does not mean, that it gives back the corrct results. First of all at the twelwth line we assign `i` to the `summe` variable, instead of incrementing it with `i`. Second in the sum we also have to the into account `n` itself, so the statements in the `while` should be less or equal to `n`.

```
1  /* Summe der ersten n Zahlen.
2   * (c) 2015 Clelia und Johannes */
3
4  #include <stdio.h>
5
6  int main () {
```

```
7       int n = 10;       /* Addiere bis zu dieser Zahl */
8       int i=0;          /* initializierung */
9       int summe=0;      /* speichert Zwischenergebnis */
10
11      while (i <= n) {
12        summe += i;     /* addiere ite Zahl auf summe */
13        i++;
14      }
15      printf ("Das Ergebnis ist %d.\n", summe);
16      return 0;
17  }
```

This can be of course faster, using the well-known closed formula for the summ of the first `n` numbers.

```
1   /* Summe der ersten n Zahlen.
2    * (c) 2015 Clelia und Johannes */
3
4   #include <stdio.h>
5
6   int main () {
7       int n = 10;       /* Addiere bis zu dieser Zahl */
8       int i=0;          /* initializierung */
9       int summe=0;      /* speichert Zwischenergebnis */
10
11      summe=n*(n+1)/2;
12      printf ("Das Ergebnis ist %d.\n", summe);
13      return 0;
14  }
```

2 Aufgabe 2

```
1  #include<stdio.h>
2  #include<math.h>//should be included (sqrt), compiled with -lm
3  int primetest(int c){
4     int n=2;
5     while  ( ((double)n<sqrt((double)c)) && (c%n != 0)){
6  //until we have not reach one divisor of c, or its square root
7  //we increase n
8        n++;
9     }
10 //if we have not found any divisor, then it is a prime
11    if (c%n != 0)
12      return 1;
13 //if we have then it is not, we give back 0
14    else
15      return 0;
16 }
17 //looking for the first prime after n
18 int firstprimeaftern(int n){
```

```
    int c=n;
//start a loop at n until the number gets a prime
    while( primetest(c) == 0 ){
//if it is not a prime yet, increase it by one
      ++c;
    }
    printf("First prime number after %d is %d\n", n, c);
    return c;
}
int main(){
    int m=31;
    int testres=primetest(m);
    int a,b,c;
    a=firstprimeaftern(20000);
    b=firstprimeaftern(30000);
    c=firstprimeaftern(40000);
    printf("%d\n", testres);
    printf("Solution b1 %d\n",a);
    printf("Solution b2 %d\n",b);
    printf("Solution b3 %d\n",c);
}
```

3 Aufgabe 3

```
//Implementing the square root
#include<stdio.h>
#include<math.h>//should be included because fabs
double sqrtown( double a){
  double x=2;
  double fx=1;
//looking for the fix point of the mapping f:0.5*(x+a/x)
//fabs(x-fx) = | x-fx | : measure how far we are from the fixpoint
  while ( fabs(x-fx)>1e-10){
     x=fx;
     fx=0.5*(x+a/x);
  }
//When we habe found it, we just print it out and give it back
  printf("Root is %f\n", x);
  return x;
}
int main(){
  double x=8.;
  double root=sqrtown(x);
  printf("Root of %e is %e\n", x, root);
}
```

4 Aufgabe 4

```c
#include<stdio.h>
int gcr( int a, int b){
//do a loop "forever"
   while (1){
//if one of the numbers get zero: we end the loop and
//return the other one: that will be the Greatest common factor.
      if (a==0) return b;
      if (b==0) return a;
//if not we make a new iteration with b, a%b, when a>b
//                              or with a, b%a in the other case
      if (a>b)
        a= a%b;
      else
        b= b%a;
   }
}
int main(){
   int a=15;
   int b=20;
   int fn=gcr(a,b);
   printf("n=%d\n",fn);
}
```

5 Aufgabe 5

You can find it in the script.

6 Aufgabe 6

```c
//we calculate an approximation to the summ of the 1/k**2 infinite series
#include<stdio.h>
#include<math.h>
int main(){
  int k=1;//we summ from 1
  double sum=0.;//very important initialization
  //we summ the one over squares till the
  //successive elements differ larger
  //then 10**-8
  while( fabs(1./(k+1)-1./(double)k)>1e-8){
    sum+=1./(double)(k*k);
    ++k;
  }
  printf("%e\n",sum);
}
```