Otto-von-Guericke-University Magdeburg
Max Planck Institute for Dynamics of Complex Technical Systems
Computational Methods in Systems and Control Theory

Dr. Jens Saak, Dipl.-Math. Martin Köhler
Website: http://www.mpi-magdeburg.mpg.de/2802004/sc1/

Winterterm 2014/2015

Scientific Computing 1
2nd Homework

Handout: 11/05/2014 **Return:** 11/12/2014

*"When reading the code in about six months and asking yourself: who wrote this crap?
The answer should not be: YOU!"*

Basically that means:

- Try to always use meaningful names for functions, variables, ...
- Write documentation wherever necessary.
- Use indentation to increase readability of the code.
- Add a short statement describing its purpose and basic behavior to each function.
- ...

Exercise 1: (5 Points)

Write a C program which computes the prime factorization of a positive integer. The integer is read from the standard input and the result is printed on the screen. If a factor occurs more than one time write it as a power. For example, the program should work like:

```
Insert a postive number: 92
The prime factorization of 92 is:
2^2 * 23^1
```

Exercise 2: (5 Points)

Design a data structure which represents a point in $\mathbb{R}^3$. On top of this structure develop the following functions:

a.) A function which reads three points from the standard input and return them as an instance of the previously defined structure via `return`.

b.) A function which reads three points from the standard input and passes the values back via a pointer in the parameter list. The return value of the function should be `void`.

c.) A function which has two points as input parameters and returns their euclidean distance.

Demonstrate all function in a small C program.
Explain the difference between how the structure is handled in Function a.) and Function b.).

Exercise 3: (6 Points)

Write a C program which analyzes some measured data. The program should behave as follows:

- The user should enter the total number of measurements.
- The program asks the user for every single measured value.
- The minimum, the maximum and the average of all values are determined and printed to the screen.

We assume that the measured values are floating point numbers.
Hint: The number of values is not known during the development or compile time.

Exercise 4: (5 Points)

We consider an array `int *f` of `n` integers. Write a program which reads the array from a file containing one integer per line. The first entry is the total number of integers to read.
Analyze the array and determine the two indices $i, j \in \{0, n-1\}, i \leq j$ such that

$$S_{ij} := \sum_{k=i}^{j} \texttt{f[k]}$$

is maximized. Think about an efficient solution.
Example data sets are available: http://www2.mpi-magdeburg.mpg.de/mpcsc/lehre/2014_WS_SC/tutorial/sum_data.tar.gz

Example: Consider the following array of length 10:

Index	0	1	2	3	4	5	6	7	8	9
Value	-1	3	4	-2	5	1	-9	4	2	-2

Then the maximum of S_{ij} is $S_{15} = 11$ beginning at $i = 1$ and ending at $j = 5$.

Exercise 5: (4 Points)

Examine the following program and try to find all memory related errors using `valgrind`. Describe the problems and their solutions. The code is also available at http://www2.mpi-magdeburg.mpg.de/mpcsc/lehre/2014_WS_SC/tutorial/demo_valgrind.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char **argv)
5  {
6          int vars[2];
7          double sum;
8          int i;
9          double data[10] = {1,22,23,344,5,34,75,36,89,540};
10         double *data2;
11
12         data2 = malloc (sizeof(double) * 10);
13
14         for (i = 0; i <= 10; i++) {
15                 data2[i]=data[i];
16         }
17
18         sum = 0.0;
19         for (i = 1; i <= 10; i++) {
20                 if ( i == 1 ) {
21                         vars[i] = data2[i];
22                 }
```

```
23                  if ( i == 2 ) {
24                          vars[i] = data2[i];
25                  }
26                  sum = sum + data2[i];
27          }
28
29
30          printf("Sum␣of␣all␣elements:␣%lg\n", sum);
31          printf("var1:␣%d␣var2:␣%d\n", vars[0], vars[1] );
32
33          return 0;
34  }
```

Hints:

- Switch the generation of debugging symbols on when you compile the program.
- Check the output of `valgrind` to get additional command line options for a deeper analysis of the problems.

Overall Points: 25