Applied Programming
Submission 1

This text will describe what to hand in for submission 1, and how this should be done. In the zip folder you can find the header files, which are named such that they refer to an exercise in the book. You need to use these and create corresponding .cpp files implementing the function prototypes specified in the header files - these should have the same name as their corresponding header file, but with the file extension *.cpp*. These are the requirements when handing in:

- All functions must be named exactly as specified in the header files
- All header and source files must be placed in a single directory and zipped as handin1.zip
- No source file are allowed to have a main function
- You are not allowed to change the existing content of the header files, but you can add something if you want to
- (to reiterate) Each header file *name.h* should have a corresponding source file *name.cpp*
- Make sure your code compiles

Using the password you received via email you can test your solution at `http://a00508.science.ku.dk:5000/` (notice the port number!). If you don't have a password or the code checker is down, then email bnq696@alumni.ku.dk. The code checker puts your submission in a queue and feedback is sent to your alumni email.

What to hand in

Exercise: **2.6**, **3.3**, **5.3**, **5.4**, **5.6**, **5.9**, **5.10**.

How to include the header file

When including e.g. the first header file for exercise 2.6, named *2_6.h*, then you simply put *#include "2_6.h"* in the top of your source file *2_6.cpp*.

In the following sections, additional descriptions or modifications of the descriptions from the book will be given.

Exercise 2.6

```
double newton_Raphson(double initialGuess, double epsilon);
```

The function should return the result, when the change is sufficiently small as explained in the book.

Exercise 3.3

```
void implicit_Euler(int n);
```

Your code should print to a file named *xy.dat* such that each line consists of the value of x, followed by a comma, followed by the value of y, i.e. *x,y*.

Exercise 5.3

```
void swap_pointer(double *a, double *b);
void swap_ref(double &a, double &b);
```

No additional description.

Exercise 5.4

```
double calc_std(double a[], int length);
double calc_mean(double a[], int length);
```

No additional description, but make sure to not divide by zero when calculating the standard deviation when the length is 1.

Exercise 5.6

```
void Multiply(double **res, double **A, double **B,
              int ARows, int ACols, int BRows, int BCols);
void Multiply(double *res, double *A, double **B,
              int ACols, int BRows, int BCols);
void Multiply(double *res, double **A, double *B,
              int ARows, int ACols, int BRows);
void Multiply(double **res, double scalar, double **B, int BRows, int BCols);
void Multiply(double **res, double **B, double scalar, int BRows, int BCols);
```

The result is written to *res*.

Exercise 5.9

```
void solve3by3(double **A, double *b, double *u);
```

The result is written to *u*.

Exercise 5.10

```
void guassian_elimination(double **A, double *b, double *u, int n);
```

The result is written to *u*. (Be aware of the miss spelling).