Project 1

Please develop the function *ScanForRoots* to scan a given function $f(x)$ for it's roots using the newton's algorithm.

The Interface

The program should read the following input parameters from an input file. The file's format is given as follows.

```
[max number of roots to find]
[lower bound: lb] [upper bound: up] [step width: sw]
[precision eps] [stepwidth h] [max iteration: ix]
```

The program should search the root of a function with the starting position x_0. The starting position x_0 should be taken from the interval which is described by the parameters **ub**, **lb** and **sw** read from the input file. The found roots of the function should be saved in an dynamical array, which should be a parameter of the function *ScanForRoots* The size of the array should be set by a parameter, which is read from the input file.

An example data file is given below.

```
10
-10. +10. 0.1
1.e-7 0.01 100
```

In the project a reasonable interval should be selected to be able to check the quality of the developed software.

The Function *ScanForRoots*

So the interface to *ScanForRoots* is the following.

```
integer function ScanForRouts(f,lb,ub,sw,eps,h,maxit,roots)
```

Parameter	Type	Comment
f	function	function to analyse
lb	real(8)	lower bound of search interval
ub	real(8)	upper bound of search interval
sw	real(8)	step width for the starting position x_0
eps	real(8)	precision for the newton algorithm
maxit	integer	maximal number of iteration for one root search
roots	real(8)	one indexed array for the found roots

The return value of *ScanForRoots* should give the number of found roots. The roots should not be stored multiple.

The main program which should call *ScanForRoots* should do the following.

- open, read and close the input file `ScanForRoots.inp`.
- call ScanForRoots[1] to search for the function's roots.
- write a little report which gives a list of the found roots

[1]It's recommended to call the already existing function `Newton` to do the job.

How to get a Project

Every student will get her/his own functions. So if you want to start with the project, please get familiar with the stuff, we have developed in our lecture and which is also available on the *info.server*. Then you should make an appointment with me, to get your very personal and special project, i.e. the testing functions which should be analyzed.

The Code

It's recommended to use parts of the code we have developed in the lecture. Please don't forget the error checking in the code. The code should be able to detect wrong input data and should also be able to handle the special cases we have discussed in the lecture. The code should be commented.

The Report

The report for this project should content a section, which describes the theory of the problem. One section should describe the usage of the developed program (like a users manual) and one section should describe the code (like a programmers guild). Here we need a description of the interfaces (parameters) of the used functions and subprograms. Algorithms should be discussed using flow charts. And don't forget the layout.

The functions of the analysis should be shown in a graph, to be able to check the programs quality.

Submission

Please note the following:

- Only pdf files for the report are accepted, so please don't submit word files or something like that!
- All projects for **CLFE** should be submitted in **one** archive file.
- Only *zip* or *rar* archive files are accepted with the following name convention
  `CLFE-[Semester]-[your name]-[the version].[zip/rar]`
- The original project sheet you have received should be inserted into the report as an appendix.
- Your name and your number have to be written onto the report's cover page.