Software lab 1 computer systems engineering

1) – simple functions, loops, control structures.

a) Write code that estimate PI.

One way to do it is with the following sum:

```
PI_est = 4 x (1 – 1/3 + 1/5 – 1/7 + 1/9 - …….)
```

Calculate `PI_est` upto 1/99.

b) Make a function from the code you wrote in a)

```
double PI_estimator ( int nMax )
```

which return `PI_est` as above with `nMax` as an argument (i.e. corresponding to 99 in example above).

c) Write a program that determines how many terms you need in the sum when you want to estimate pi with a precision: ± 1% of 3.14159.

2) – Simple functions, loops and control statements.
In Appendix 1 100 numbers are listed which you will put in a vector. Write functions that calculate the following properties from the vector:

1) Average
2) Minimum value
3) Maximum value
4) Variance

Average
Write the function `average(..)` so that it calculates the average of the numbers in `dVec` return it. The argument `nVals` states how many number that are contained in `dVec`.

```
double average(double dVec[], int nVals);
```

hint:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}(x_1 + \cdots + x_n)$$

Minimum value
Write the function `minima` (..) that finds the smallest number in the vector `dVec`. The argument `nVals` states how many number that are contained in `dVec`.

```
double minima(double dVec[], int nVals);
```

Maximum value
Write the function `maxima` (..) that finds the largest number in the vector `dVec`. The argument `nVals` states how many number that are contained in `dVec`.

```
double maxima(double dVec[], int nVals);
```

Variance
Write the function `variance(..)` so that it calculates the variance of the numbers in `dVec` return it. The argument `nVals` states how many number that are contained in `dVec`. If it is not possible to calculate the variance, eg. there is one value in the vector, return -1.0;

```
double variance(double dVec[], int nVals);
```

Hint:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N} (x_i - \overline{x})^2$$

Where xi is the individual elements and xbar is the average defined above.

3) – Simple functions, loops, control statements, arrays, data types, size of data type in memory.

a) Create a function

```
stats_vars calc_stats(double dVec[], int nVals);
```

where

`stats_vars`

is your own data type with the following members:

1) Number of data in the vector
2) average value
3) minminimum value
4) maximum value
5) variance

Hint: typedef and struct.

Test the function with the numbers provided in appendix.

b) Check the size of the types.
Hint: Nyckelord är sizeof.
Compare if the size of the sum of individual members are the same as for `stats_vars`

General hints:

There is a function for calculating x squared.
pow(x, y)
where x is the value you raise to y (eg 2 for the square).
To use the math functions in C add this statements at the top of the source code file:
#include <math.h>

APPENDIX 1:

0.9501, 0.2311, 0.6068, 0.4860, 0.8913, 0.7621, 0.4565, 0.0185, 0.8214, 0.4447, 0.6154, 0.7919, 0.9218, 0.7382, 0.1763, 0.4057, 0.9355, 0.9169, 0.4103, 0.8936, 0.0579, 0.3529, 0.8132, 0.0099, 0.1389, 0.2028, 0.1987, 0.6038, 0.2722, 0.1988, 0.0153, 0.7468, 0.4451, 0.9318, 0.4660, 0.4186, 0.8462, 0.5252, 0.2026, 0.6721, 0.8381, 0.0196, 0.6813, 0.3795, 0.8318, 0.5028, 0.7095, 0.4289, 0.3046, 0.1897, 0.1934, 0.6822, 0.3028, 0.5417, 0.1509, 0.6979, 0.3784, 0.8600, 0.8537, 0.5936, 0.4966, 0.8998, 0.8216, 0.6449, 0.8180, 0.6602, 0.3420, 0.2897, 0.3412, 0.5341, 0.7271, 0.3093, 0.8385, 0.5681, 0.3704, 0.7027, 0.5466, 0.4449, 0.6946, 0.6213, 0.7948, 0.9568, 0.5226, 0.8801, 0.1730, 0.9797, 0.2714, 0.2523, 0.8757, 0.7373, 0.1365, 0.0118, 0.8939, 0.1991, 0.2987, 0.6614, 0.2844, 0.4692, 0.0648,
0.9883