# Assignment 4

## CSCI 2132: Software Development

Due March 4, 2019

---

Assignments are due on the due date before 23:59. All assignments must be submitted electronically via the course SVN server. Plagiarism in assignment answers will not be tolerated. By submitting your answers to this assignment, you declare that your answers are your original work and that you did not use any sources for its preparation other than the class notes, the textbook, and ones explicitly acknowledged in your answers. Any suspected act of plagiarism will be reported to the Faculty's Academic Integrity Officer and possibly to the Senate Discipline Committee. The penalty for academic dishonesty may range from failing the course to expulsion from the university, in accordance with Dalhousie University's regulations regarding academic integrity.

## General Instructions: How to Submit Your Work

You must submit your assignment answers electronically:

- Change into your subversion directory on bluenose: `cd ~/csci2132/svn/CSID`.
- Run `svn update`. This will create a directory `a4` in your subversion directory. Inside this directory, you will find a file `genseq.c`.
- Change into your assignment directory: `cd a4`.
- Compile the `genseq.c` file to executable code using `gcc -o genseq genseq.c`. You will need this program for the second question.
- Create files inside the `a4` directory as instructed in the questions below and put them under Subversion control using `svn add <filename>`. **Only add the files you are asked to add!**
- Once you are done answering all questions in the assignment (or the ones that you are able to answer—hopefully all), the contents of your `a4` directory should look like this:

```
a4
├── genseq.c
├── primetest.c
└── stats.c
```

(You will also have executable programs and potentially some data files in this directory, but you should not add them to SVN.) Submit your work using `svn commit -m"Submit Assignment 4"`.

## (Q1) Primality Testing — 10 marks

Recall that a number is a prime number if its only divisors are 1 and itself. Write a C program that prints "Enter a number: " to `stdout` and reads an unsigned integer from `stdin`. (Use `scanf` to do so. You do not need to perform any error checks.) Your program should print "$x$ is a prime number.", where $x$ is the number you entered, if $x$ is a prime number. Otherwise, it should print "$x$ is not a prime number." The exit code of your program should be 0.

For full marks, your program must meet the following requirements concerning the implementation:

- You should be able to handle any unsigned integer that fits in 64 bits.
- When testing whether there exists a number $y$ that divides $x$ (and, thus, $x$ is not a prime number), restrict yourself to integers $y \leq 2\sqrt{x}$. Precisely, the largest number $y$ you test whether it divides $x$ should be between $\sqrt{x}$ and $2\sqrt{x}$. The logic is that, if there exists a number $z \geq \sqrt{x}$ that divides $x$, then $x = yz$ for some number $y \leq \sqrt{x}$ and $y$ also divides $x$. The reason why the permissible range of $y$ is $[2, 2\sqrt{x}]$ is explained in the next bullet.
- For full marks, find the maximum value of $y$ you test without calling the `sqrt` library function (which is a floating point function). Instead, use C's bit shift operator to calculate the maximum value of $y$ you need to test. This may return a number as large as $2\sqrt{x}$.

Implement your program in a file `primetest.c`, place this file into your `a4` directory and commit it to SVN.

## (Q2) Calculating the Distribution of Values — 10 marks

For a set of $n$ real numbers $\{x_1, \ldots, x_n\}$, here are a number of basic statistics we would like to know about the distribution of these numbers:

- The minimum value $m = \min\{x_1, \ldots, x_n\}$.
- The maximum value $M = \max\{x_1, \ldots, x_n\}$.
- The average value $a = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- The standard deviation $s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - a)^2}$.

Write a program `stats.c` that calculates these values for a sequence of values it reads from `stdin`.

The input is provided as a sequence of $n+1$ numbers, each on a separate line. The first number is $n$. The remaining $n$ numbers are $x_1, \ldots, x_n$. Your program may assume that $n \leq 1000$. For two bonus marks, your code should be able to handle any value of $n$, even values greater than 1000.

For any given input, your program should print the output

```
m = ...
M = ...
a = ...
s = ...
```

to `stdout`, where the `...` are to be replaced with the values of $m$, $M$, $a$, and $s$. Each output value should be represented with 3 digits after the decimal point.

To generate some test data for your program, you can use the `genseq` program you created earlier. The invocation `./genseq <n>` generates a sequence of $n$ random numbers and prints $n$ followed by these $n$ random numbers to `stdout`. Using `./genseq <n> > seq.txt`, you can save this output in a file `seq.txt`. You can use this as the test input for your `stats` program by running `./stats < seq.txt`.

Add your file `stats.c` to SVN and commit it.