THE UNIVERSITY OF SYDNEY

COMP2129 Assignment 3

Due: 9:00pm Sunday, 15 May 2016

This assignment is worth 8% of your final assessment

Task description

In this assignment we will implement a matrix computation engine that is able to construct matrices and perform matrix computations in the C programming language. The aim is to use a variety of parallel programming, code optimisation and algorithmic techniques to achieve peak performance.

You have been provided with some well documented scaffold code. Your task is to implement the incomplete functions and then analyse the running time of the operations under various inputs with an aim to make each operation run as fast as possible. Most of the matrix operations can be improved by using parallelisation, more efficient algorithms and code optimisation techniques. It is up to you to determine whether you will focus on tuning certain operations that are dramatically slower than the others, or to focus on each operation equally. Make sure to cache matrices and results where possible.

You are encouraged to ask questions on Ed using the assignments category. As with any assignment, make sure that your work is your own, and you do not share your code or solutions with other students.

Working on your assignment

You can work on this assignment on your own computer or the lab machines. Students can also take advantage of the online code editor on Ed. Simply navigate to the assignment page and you are able to run, edit and submit code from within your browser. We recommend that you use Safari or Chrome.

It is important that you continually back up your assignment files onto your own machine, flash drives, external hard drives and cloud storage providers. You are encouraged to submit your assignment while you are in the process of completing it. By submitting you will obtain some feedback of your progress.

Academic declaration

By submitting this assignment you declare the following:

I declare that I have read and understood the University of Sydney Academic Dishonesty and Plagiarism in Coursework Policy, and except where specifically acknowledged, the work contained in this assignment or project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Academic Dishonesty and Plagiarism in Coursework Policy, can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.

I acknowledge that the School of Information Technologies, in assessing this assignment, may reproduce it entirely, may provide a copy to another member of faculty, and or communicate a copy of this assignment to a plagiarism checking service or in house computer program, and that a copy of the assignment may be maintained by the service or the School of IT for the purpose of future plagiarism checking.

Implementation details

Your task is to implement a matrix computation engine based on the included scaffold code. You can assume only valid input will be tested. All matrices will be square and have the same dimensions between $1 \leqslant order \leqslant 10,000$. All matrix rows, columns and elements will be indexed from 1.

Your program must be contained in `matrix.c` and `matrix.h` and produce no errors when built and run on the lab machines and Ed with the following command:

```
$ clang -O1 -std=gnu11 -Wall -Werror -pthread main.c matrix.c -o matrix
```

Your program output must match the exact output format shown in the examples and on Ed. You are encouraged to submit your assignment while you are working on it, so you can obtain some feedback.

Commands

```
SET <key> = identity
SET <key> = random <seed>
SET <key> = uniform <value>
SET <key> = sequence <start> <step>

SET <key> = cloned <matrix>
SET <key> = sorted <matrix>
SET <key> = rotated <matrix>
SET <key> = reversed <matrix>
SET <key> = transposed <matrix>

SET <key> = scalar.add <matrix> <scalar>
SET <key> = scalar.mul <matrix> <scalar>
SET <key> = matrix.add <matrix a> <matrix b>
SET <key> = matrix.mul <matrix a> <matrix b>
SET <key> = matrix.pow <matrix> <exponent>
SET <key> = matrix.conv <matrix> <kernel>

SHOW <key>
SHOW <key> row <number>
SHOW <key> column <number>
SHOW <key> element <row> <column>

COMPUTE sum <key>
COMPUTE trace <key>
COMPUTE minimum <key>
COMPUTE maximum <key>
COMPUTE determinant <key>
COMPUTE frequency <key> <value>
```

Examples

```
> SET a = identity
ok

> SHOW a
1.00 0.00
0.00 1.00

> SET b = random 1
ok

> SHOW b
41.00 18467.00
6334.00 26500.00

> SET c = random 2
ok

> SHOW c
45.00 29216.00
24198.00 17795.00

> SET d = uniform 0
ok

> SHOW d
0.00 0.00
0.00 0.00

> SET e = uniform 1
ok

> SHOW e
1.00 1.00
1.00 1.00

> SET f = sequence 1 1
ok

> SHOW f
1.00 2.00
3.00 4.00

> SET g = sequence 1 2
ok

> SHOW g
1.00 3.00
5.00 7.00
```

```
> SET a = identity
ok

> SHOW a
1.00 0.00
0.00 1.00

> SET b = sequence 1 1
ok

> SHOW b
1.00 2.00
3.00 4.00

> SET c = scalar.add a 1
ok

> SHOW c
2.00 1.00
1.00 2.00

> SET d = scalar.add b 4
ok

> SHOW d
5.00 6.00
7.00 8.00

> SET d = scalar.mul a 1

> SHOW d
1.00 0.00
0.00 1.00

> SET e = scalar.mul c 2
ok

> SHOW e
4.00 2.00
2.00 4.00

> SET f = scalar.mul e 2
ok

> SHOW f
8.00 4.00
4.00 8.00
```

```
> SET a = identity
ok

> SET b = uniform 4
ok

> SET c = sequence 1 1
ok

> SHOW c
1.00 2.00
3.00 4.00

> SET d = matrix.add b c
ok

> SHOW d
5.00 6.00
7.00 8.00

> SET e = matrix.mul c a
ok

> SHOW e
1.00 2.00
3.00 4.00

> SET f = matrix.mul c d
ok

> SHOW f
19.00 22.00
43.00 50.00

> SET g = matrix.pow c 0
ok

> SHOW h
1.00 0.00
0.00 1.00

> SET i = matrix.pow c 4
ok

> SHOW i
199.00 290.00
435.00 634.00
```

```
> SET a = identity
ok

> SHOW a
1.00 0.00
0.00 1.00

> SET b = uniform 4
ok

> SHOW b
4.00 4.00
4.00 4.00

> COMPUTE sum a
2.00

> COMPUTE sum b
16.00

> COMPUTE trace a
2.00

> COMPUTE trace b
8.00

> COMPUTE minimum a
0.00

> COMPUTE minimum b
4.00

> COMPUTE maximum a
1.00

> COMPUTE maximum b
4.00

> COMPUTE determinant a
1.00

> COMPUTE determinant b
0.00

> COMPUTE frequency a 1
2

> COMPUTE frequency a 2
0
```

Writing your own testcases

We have provided you with some test cases but these do not test all the functionality described in the assignment. It is important that you thoroughly test your code by writing your own testcases.

You should place all of your testcases in a `tests/` folder. Ensure that each testcase has a `.in` input file along with a corresponding `.out` output file. We recommend that the names of your testcases are descriptive so that you know what each is testing, e.g. `sum.in` and `trace.in`

Submission details

Final deliverable for the performance is due in week 10. Further details will be announced on Ed.

You must submit your code using the assignment page on Ed. To submit, simply place your files and folders into the workspace, click run to check your program works and then click submit.

You are encouraged to submit multiple times, but only your last submission will be marked.

Marking

In this assignment you will receive marks for correctness and performance of your program. However, submissions are only eligible for the performance component if they pass all of the correctness tests.

5 **marks** are assigned based on automatic tests for the *correctness* of your program.
This component will use our own hidden test cases that cover every aspect of the specification. To pass test cases your solution must produce the correct output within the imposed time limit.

5 **marks** are assigned based on the *performance* of your code relative to the benchmark.
This component is tested on a separate machine in a consistent manner. Submissions faster than the benchmark implementation will receive 5 marks, with successively slower ones receiving lower marks. Submissions faster than our basic implementation will receive at least 2.5 marks.

Your program will be marked automatically, so make sure that you carefully follow the assignment specifications. Your program must match the exact output in the examples and the testcases on Ed.

Warning: Any attempts to deceive or disrupt the marking system will result in an immediate zero for the entire assignment. Negative marks can be assigned if you do not properly follow the assignment specification, or your code is unnecessarily or deliberately obfuscated.