All Pairwise Computation is defined as performing computation (e.g., correlations) between every pair of the elements in a given dataset. In this assignment, we consider a set of N sequences, each being of length M, which are stored in a two-dimensional (2D) matrix of size N by M (i.e., N rows and M columns). We then calculate a dot product for every possible pair of sequences (row vectors) in the matrix.

A dot product of two sequences, or vectors is defined as

$$x \cdot y = \sum_{i=0}^{M-1} x_i \, y_i$$

If we allow a vector to pair with itself, then there are $N(N+1)/2$ such pairs in total for N vectors. Assume N is equal to 5 for example. We then have the following 15 vector pairs:

(0, 0)(0, 1)(0, 2)(0, 3)(0, 4)

(1, 1)(1, 2)(1, 3)(1, 4)

(2, 2)(2, 3)(2, 4)

(3, 3)(3, 4)

(4, 4)

In the above dot product formula, (i, j) denotes the pair of sequences i and j. After the dot product computation for pair (i, j), we will have a single value as the output. Let a_i and a_j be two sequences, namely the i^{th} and j^{th} row vectors in the 2D matrix, and $c_{ij} = a_i \cdot a_j$. We can store the computational results in a one-dimensional (1D) array of size $N(N+1)/2$ in a row major order:

c_{00}	c_{01}	c_{02}	c_{03}	c_{04}	c_{11}	c_{12}	c_{13}	c_{14}	c_{22}	c_{23}	c_{24}	c_{33}	c_{34}	c_{44}

In this assignment you are asked to design an **efficient** parallel algorithm to solve the above problem on a shared-memory computing platform and implement the algorithm using Pthreads.

You need to first implement another simple parallel algorithm as a baseline for performance comparison. This baseline algorithm is described as follows.

The baseline parallel algorithm: It is easy to see that the computation of dot products for different sequence pairs can be done independently since there is no data dependency. We can simply partition the resulting 1D array into T groups and assign one group to each thread assuming the number of threads created to solve the problem is T. Assuming $N = 5$ and $T = 4$, there are 15 different pairs (as shown in the above example) and each thread will be assigned 4 pairs for dot product computation except the last one which is given 3 pairs:

thread 0: c_{00}, c_{01}, c_{02}, c_{03}
thread 1: c_{04}, c_{11}, c_{12}, c_{13}
thread 2: c_{14}, c_{22}, c_{23}, c_{24}
thread 3: c_{33}, c_{34}, c_{44}

In your baseline implementation, you must seriously consider load balancing; i.e., the pairs are evenly distributed among all threads and the difference is no greater than one.

After you successfully implement the baseline algorithm using Pthreads, you can design and implement your own **efficient** parallel algorithm using Pthreads. In your design, you **must** consider how to use techniques such as blocking and loop unrolling to improve the performance.

In your "efficient parallel" implementation:

- Your program needs to ask for *N*, *M* and *T* as user defined parameters (and block size *B* if necessary);
- The results must be placed in a 1D array in the row major order as shown in the above example; and
- After the parallel computation, your main program must conduct a self-check for correctness; i.e., perform a sequential computation using the same data set and then compare the result of this computation with the parallel computation result.

Both implementations must be in C and Pthread.

You **must** write a **report**. The report must be concise, clear (3-6 A4 pages) and contain the following sections:

1. Problem definition and requirements
2. Parallel algorithm design and implementation
3. Testing and performance evaluation
4. Discussion
5. Known issues in program
6. Manual (e.g. how to run the program, input and output)

When discussing performance evaluation in your report, you must draw at least two figures to compare the performance between the baseline algorithm and your own algorithm. One figure illustrates the execution times of computation using different problem sizes for a fixed number of cores. The other illustrates the execution times of computation using different number of cores for a problem of fixed size.

Your assignments will be marked on the efficiency of your algorithm, program logic and readability, accuracy of results, and quality of your report.

You MUST attempt this assignment **individually**.

Submission Requirements

1. Your submission must be made by **11:59pm on Friday, 8 April, 2022 (Sydney time)**.
2. Create **a tar or zip file** that contains **your report**, ***makefile*** and **source files** (e.g., **.c** and **.h files**). **DO NOT INCLUDE ANY OBJECT OR BINARY FILES.**
3. Submit only one **.tar** or **.zip file**.

Failure to follow these submission requirements may lead to loss of marks.