1 Problem Description

This is the laboratory exercise that requires to write algorithm that calculates the transpose of a square matrix of specified size, A[N0][N1] where N0 = N1. N0=128, 1024, 2048, and 4096. The written algorithm is not limited to these values and smaller sizes as well as larger numbers can be tested to evaluate the performance of the algorithm. Here, the basic idea is to have the elements of the input matrix reshuffled using a small temporal storage space much less than N0XN1, to generate the transposed elements in the input matrix. In the algorithm, the number of threads are chosen assigned by the user before the algorithm is compiled and run. More importantly, for each value of N0 and the number of threads used, time it takes to compute the transposition of a matrix is evaluated excluding the time it takes to populate the input matrix.

2 Solution

2.1 Solution Description

As usual, various directives and headers relevant to execute the program in C are specified including omp.h and the system timer for calculating the time taken by the algorithm to execute the task at hand.

Various variables are defined as well with the likes of input matrix, number of threads, number of rows and cols and time-related variables all necessary for the successful compilation and running of the program. Memory was dynamically assigned both input matrix and transposed matrices with the help of malloc (memory allocation) function.

The input matrix was first initialized and then populated with some numbers to begin with and the transposed matrix was also initialized to zeros before being able to populate it with the reshuffled elements of the input matrix. The algorithm makes use of both serial and parallel computation of the transposition of the matrix and timer was set up to evaluate the time it takes for the program to execute after which both the input matrix and the transposed matrix are displayed as well as the time the program took to compute the the transposition.

2.2 Pseudocode

The pseudocode is described here below:

Algorithm 1: Algorithm Pseudocode

```
include header files
main program start
define variables
check for command line args
for i = 0; i < nRows; i = i + 1 do
    Mat = (float *) malloc(sizeof(float) * nCols); for
      j = 0; j < nCols; j = j + 1 do
        M
    end
    at[i][j];
end
initialize transpose matrix to 0
get time
start parallel execution of the code
input matrix elements are reshuffled to transpose matrix
Trans[j][i] = Mat[i][j]
all threads join Master thread
compute transposition of matrix serially
display the input matrix
display the transposed matrix
free allocated memory
```