UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Part - 1 : High Performance Computing

Student Id	: 2407710
Student Name	: Dhadkan K.C.
Group	: L6CG4
Lecturer	: Mr.Yamu Poudel

Table of Contents

Task – 1

Word Occurrence Counting using Multithreading (Pthread)

The task is aimed at writing a multithreaded C program with the help of Pthread library to enumerate the number of times words in a text file appear. The input file name and the number of threads are the command-line arguments that the program takes. The biggest goal is to enhance the speed of the processing by sharing the work load across several threads but is that the results are correct by ensuring proper synchronization.

The program starts by reading the input file line by line and dynamically storing these lines in the memory. These lines are separated parts of work which can be generated at the same time. Rather than simply assigning lines to threads on a static basis, a dynamic allocation system is provided wherein a thread loads in the next available line by means of a shared index. This will provide enhanced load balancing, in cases where the length of the lines are not equal.

The threads read through their respective lines one character at a time. Alphabetic letters are clustered together to create words, marks of punctuation are not taken into account and all characters have been turned into lower-case. This normalization can make the words like Word, word and WORD the treatment of a single entity. When a word is created, it is included in a common word list in which its frequency will be updated

Mutex lock is employed to eliminate race conditions. The mutex guards access to the shared line index in order to ensure that two threads do not work on the same line. The shared word and frequency arrays are another mutex that guarantees that updates to word counts can be made across multiple threads that are trying to update the word counts at the same time.

The management of memories is done attentively during the program. The file lines are stored using dynamic allocation, and all the memory allocated is released appropriately upon the termination of the program. Error checking is also provided on file opening, memory allocation, and thread creation so that it can be executed robustly.

Once all the threads have been executed, the words are arranged in a descending order (according to their frequency) and sorted by the program. The resulting whole line is saved in an output file which is called result.txt with the word preceded by the number of times it appears.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <ctype.h>

#define WORD_LEN 50
#define MAX_WORDS 10000

char **lines;
int total_lines = 0;
int current_line = 0;

char words[MAX_WORDS][WORD_LEN];
int counts[MAX_WORDS];
int total_words = 0;

pthread_mutex_t line_mutex;
pthread_mutex_t word_mutex;
```

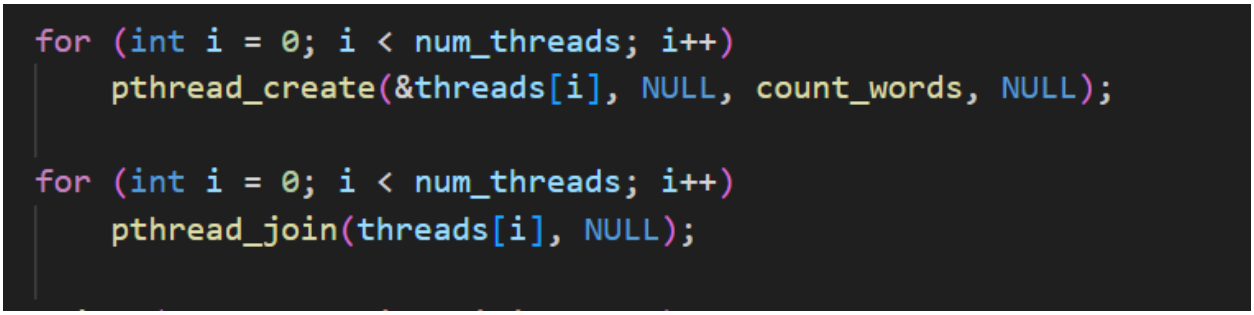
This section includes the required libraries, sets constants for maximum word length and total words, and declares all global variables. Arrays lines, words, and counts store the text data and word frequencies. Mutexes ensure safe access to shared resources when multiple threads run concurrently.

Main Function

```c
int main(int argc, char *argv[])
{
    if (argc != 3) {
        printf("Usage: %s <file> <threads>\n", argv[0]);
        return 1;
    }

    FILE *fp = fopen(argv[1], "r");
    if (!fp) {
        printf("Cannot open file\n");
        return 1;
    }

    int num_threads = atoi(argv[2]);
    pthread_t threads[num_threads];

    printf("Word count program starting...\n");
    printf("Input file: %s\n", argv[1]);
    printf("Number of threads: %d\n", num_threads);
    printf("Output file: result.txt\n\n");

    pthread_mutex_init(&line_mutex, NULL);
    pthread_mutex_init(&word_mutex, NULL);

    int capacity = 100;
    lines = malloc(capacity * sizeof(char *));
    char buffer[256];

    while (fgets(buffer, sizeof(buffer), fp)) {
        if (total_lines == capacity) {
            capacity *= 2;
            lines = realloc(lines, capacity * sizeof(char *));
        }
        lines[total_lines] = strdup(buffer);
        total_lines++;
    }

    fclose(fp);

    printf("Total lines read: %d\n", total_lines);
    printf("Starting word counting...\n");
```

The principal role begins by checking the command-line parameters. The program will be given two arguments namely the input file name and the number of threads. In case the arguments are wrong, then a usage message is displayed and the program is terminated. The input file is then read in read mode and the execution halts in case the file cannot be accessed. The thread count is transformed into string and then into an integer number which is used to form an array of thread identifiers.

This section includes the required libraries, sets constants for maximum word length and total words, and declares all global variables. Arrays lines, words, and counts store the text data and word frequencies. Mutexes ensure safe access to shared resources when multiple threads run concurrently.



```
for (int i = 0; i < num_threads; i++)
    pthread_create(&threads[i], NULL, count_words, NULL);

for (int i = 0; i < num_threads; i++)
    pthread_join(threads[i], NULL);
```

threads are created to process lines in parallel. Each thread runs the count_words function. pthread_join ensures that the main function waits until all threads have finished counting before proceeding.

```c
void add_word(char *word)
{
    pthread_mutex_lock(&word_mutex);

    for (int i = 0; i < total_words; i++) {
        if (strcmp(words[i], word) == 0) {
            counts[i]++;
            pthread_mutex_unlock(&word_mutex);
            return;
        }
    }

    strcpy(words[total_words], word);
    counts[total_words] = 1;
    total_words++;

    pthread_mutex_unlock(&word_mutex);
}

void *count_words(void *arg)
{
    while (1) {
        pthread_mutex_lock(&line_mutex);
        int line_number = current_line++;
        pthread_mutex_unlock(&line_mutex);

        if (line_number >= total_lines)
            break;

        char *line = lines[line_number];
        char word[WORD_LEN];
        int idx = 0;

        for (int i = 0; line[i]; i++) {
            if (isalpha(line[i])) {
                word[idx++] = tolower(line[i]);
            } else if (idx > 0) {
                word[idx] = '\0';
                add_word(word);
                idx = 0;
            }
        }

        if (idx > 0) {
            word[idx] = '\0';
            add_word(word);
        }
    }
    return NULL;
}
```

These functions handle word extraction and counting. count_words runs in each thread and processes one line at a time. add_word adds a new word or increments the frequency, with mutex locking to avoid conflicts between threads.

```c
void sort_by_frequency()
{
    for (int i = 0; i < total_words - 1; i++) {
        for (int j = i + 1; j < total_words; j++) {
            if (counts[j] > counts[i]) {
                int tmp_count = counts[i];
                counts[i] = counts[j];
                counts[j] = tmp_count;

                char tmp_word[WORD_LEN];
                strcpy(tmp_word, words[i]);
                strcpy(words[i], words[j]);
                strcpy(words[j], tmp_word);
            }
        }
    }
}
```

After counting is complete, the words are sorted by frequency and written to result.txt. This ensures the most frequent words appear first, and the program produces a clean, readable output.

Compiling

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ ./1 WordOccurrenceDataset.txt 4
Word count program starting...
Input file: WordOccurrenceDataset.txt
Number of threads: 4
Output file: result.txt

Total lines read: 120000
Starting word counting...
Word counting finished
Sorting results...
Processing completed
Total unique words: 94
Results written to result.txt
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$
```

InputFIle

```
parallel
university
compilation
algorithm
function
abstraction
database
operation
storage
teacher
computer
semantic
tree
recursion
kernel
encryption
cloud
security
student
research
engineering
validation
assignment
validation
computer
feedback
CUDA
linkedlist
feedback
efficiency
subtraction
tree
```

Output Result.txt

```
student : 2456
teacher : 2413
linkedlist : 1328
synchronisation : 1323
assessment : 1318
complexity : 1317
network : 1317
pthread : 1315
recursion : 1315
function : 1313
security : 1309
computer : 1309
runtime : 1308
array : 1307
learning : 1290
searching : 1286
sorting : 1285
syntax : 1285
python : 1282
queue : 1281
output : 1281
semantic : 1278
object : 1278
efficiency : 1277
class : 1277
loop : 1274
matrix : 1272
storage : 1268
interface : 1268
logic : 1267
inclusive : 1267
```

Task 2

Multiple Operations with Matrices using Multithreading (OpenMP)

This assignment is based on the creation of an OpenMP based C program that can use multiple processors to perform different operations on matrices. The program reads many matrices in a text file that is given through command line and it dynamically stores them in memory. Matrices are manipulated in pairs, and all the valid operations including addition, subtraction, multiplication and division of elements, transpose, and conventional multiplication of matrices are possible, depending on the dimensions of the pair of matrices.

The program gives the user an option of the number of threads to use. OpenMP instructions, specifically the use of the omp parallel for directive, are used to parallelise the most CPU intensive processes, e.g., when looping through rows and columns to carry out element wise processes or even matrix multiplication. In order to remain efficient, the number of threads is dynamically limited depending on the size of the matrices and over-threading can be avoided with a large number of threads compared to work items.

Such invalid operations like adding matrices of dissimilar dimensions or multiplying matrices of different sizes are treated in a graceful manner with easy to understand messages in the output file. All the findings are captured in a readable and structured data format and the dynamically allocated memory is freed adequately to avoid any memory leakages.

The most significant advantage of this implementation is that the organization uses OpenMP parallelization intelligently, so that the operations with matrices are performed effectively without losing their correctness and stability based on a variety of input files and different numbers of threads.

Main Function

```c
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>
#include <math.h>

double **allocMatrix(int r, int c) {
    double **m = malloc(r * sizeof(double *));
    for (int i = 0; i < r; i++)
        m[i] = malloc(c * sizeof(double));
    return m;
}

void freeMatrix(double **m, int r) {
    for (int i = 0; i < r; i++)
        free(m[i]);
    free(m);
}

void printMatrix(FILE *f, double **m, int r, int c) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            if (isnan(m[i][j]))
                fprintf(f, "NaN");
            else
                fprintf(f, "%lf", m[i][j]);
            if (j < c - 1) fprintf(f, ", ");
        }
        fprintf(f, "\n");
    }
}
```

This section sets up the necessary libraries for matrix operations and OpenMP parallelization. Functions allocMatrix and freeMatrix handle dynamic allocation and deallocation of matrices, allowing matrices of any size to be processed safely.

```c
void printMatrix(FILE *f, double **m, int r, int c) {
    for (int i = 0; i < r; i++) {
        for (int j = 0; j < c; j++) {
            if (isnan(m[i][j]))
                fprintf(f, "NaN");
            else
                fprintf(f, "%lf", m[i][j]);
            if (j < c - 1) fprintf(f, ", ");
        }
        fprintf(f, "\n");
    }
}
int capThreads(int req, int max)
int capThreads(int req, int max) {
    return req < max ? req : max;
}
```

printMatrix prints matrices into the output file, using "NaN" for invalid values. capThreads ensures that requested threads do not exceed the number of rows, which optimizes OpenMP parallelization.

```c
double **add(double **A, double **B, int r, int c, int t) {
    double **R = allocMatrix(r, c);
    t = capThreads(t, r);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            R[i][j] = A[i][j] + B[i][j];
    return R;
}

double **sub(double **A, double **B, int r, int c, int t) {
    double **R = allocMatrix(r, c);
    t = capThreads(t, r);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            R[i][j] = A[i][j] - B[i][j];
    return R;
}

double **elemMul(double **A, double **B, int r, int c, int t) {
    double **R = allocMatrix(r, c);
    t = capThreads(t, r);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            R[i][j] = A[i][j] * B[i][j];
    return R;
}

double **elemDiv(double **A, double **B, int r, int c, int t) {
    double **R = allocMatrix(r, c);
    t = capThreads(t, r);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            R[i][j] = (B[i][j] == 0) ? NAN : A[i][j] / B[i][j];
    return R;
}

double **transpose(double **A, int r, int c, int t) {
    double **T = allocMatrix(c, r);
    t = capThreads(t, r);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < r; i++)
        for (int j = 0; j < c; j++)
            T[j][i] = A[i][j];
    return T;
}

double **matMul(double **A, double **B, int rA, int cA, int cB, int t) {
    double **R = allocMatrix(rA, cB);
    t = capThreads(t, rA);
    #pragma omp parallel for num_threads(t)
    for (int i = 0; i < rA; i++) {
        for (int j = 0; j < cB; j++) {
            double sum = 0;
            for (int k = 0; k < cA; k++)
                sum += A[i][k] * B[k][j];
```

These functions implement addition, subtraction, element-wise multiplication/division, transpose, and matrix multiplication. All heavy loops are parallelized with OpenMP (#pragma omp parallel for) to use multiple threads efficiently. capThreads ensures thread count is safe for small matrices.

```c
int main(int argc, char *argv[]) {
    if (argc != 3) {
        printf("Usage: ./matrix <file> <threads>\n");
        return 1;
    }

    FILE *in = fopen(argv[1], "r");
    if (!in) {
        printf("Input file error\n");
        return 1;
    }

    int threads = atoi(argv[2]);
    FILE *out = fopen("Task2_MatData_6.txt", "w");


    printf("Matrix operations program starting...\n");
    printf("Input file: %s\n", argv[1]);
    printf("Number of threads: %d\n", threads);
    printf("Output file: Task2_Matrices_12.txt\n\n");


    int pair = 1;

    while (1) {
        int r1, c1, r2, c2;
        if (fscanf(in, "%d,%d", &r1, &c1) != 2) break;

        double **A = allocMatrix(r1, c1);
        for (int i = 0; i < r1; i++)
            for (int j = 0; j < c1; j++)
                fscanf(in, "%lf,", &A[i][j]);

        if (fscanf(in, "%d,%d", &r2, &c2) != 2) break;

        double **B = allocMatrix(r2, c2);
        for (int i = 0; i < r2; i++)
            for (int j = 0; j < c2; j++)
                fscanf(in, "%lf,", &B[i][j]);
```

The main loop reads pairs of matrices from the input file. Dynamic memory is allocated for each matrix, and the program will continue until the end of the file. Each pair is then processedfor all operations.

The main loop reads pairs of matrices from the input file. Dynamic memory is allocated for each matrix, and the program will continue until the end of the file. Each pair is then

```
    if (r1 == r2 && c1 == c2) {
        fprintf(out, "Addition (%d,%d)\n", r1, c1);
        double **R = add(A, B, r1, c1, threads);
        printMatrix(out, R, r1, c1);
        freeMatrix(R, r1);

        fprintf(out, "\nSubtraction (%d,%d)\n", r1, c1);
        R = sub(A, B, r1, c1, threads);
        printMatrix(out, R, r1, c1);
        freeMatrix(R, r1);

        fprintf(out, "\nElement-wise Multiply (%d,%d)\n", r1, c1);
        R = elemMul(A, B, r1, c1, threads);
        printMatrix(out, R, r1, c1);
        freeMatrix(R, r1);

        fprintf(out, "\nElement-wise Divide (%d,%d)\n", r1, c1);
        R = elemDiv(A, B, r1, c1, threads);
        printMatrix(out, R, r1, c1);
        freeMatrix(R, r1);
    } else {
        fprintf(out, "Addition cannot be done (different size)\n");
        fprintf(out, "Subtraction cannot be done (different size)\n");
        fprintf(out, "Element-wise operations cannot be done (different size)\n");
    }

    fprintf(out, "\nTranspose A (%d,%d)\n", c1, r1);
    double **T = transpose(A, r1, c1, threads);
    printMatrix(out, T, c1, r1);
    freeMatrix(T, c1);

    fpri double **transpose(double **A, int r, int c, int t)
    T = transpose(B, r2, c2, threads);
    printMatrix(out, T, c2, r2);
    freeMatrix(T, c2);

    if (c1 == r2) {
        fprintf(out, "\nMatrix Multiply (%d,%d)\n", r1, c2);
        double **R = matMul(A, B, r1, c1, c2, threads);
        printMatrix(out, R, r1, c2);
        freeMatrix(R, r1);
    } else {
        fprintf(out, "\nMatrix multiplication cannot be done\n");
    }

    freeMatrix(A, r1);
    freeMatrix(B, r2);
}


printf("\nProcessing completed. %d matrix pairs processed.\n", pair - 1);
printf("Results written to Task2_Matrices_12.txt\n");


fclose(in);
fclose(out);
return 0;
```

In this case, the program does every possible operation of all pairs of matrices:

Addiction, subtraction, element-wise multiply/divide - only on same sized matrices. Transpose - always possible.

Multiplication of matrices - it is only possible when the columns of A and the rows of B are equal.

In case the dimensions are not matched, the program displays messages such as, the multiplication of two matrixes is not possible. This is why your run with Matrices.txt provokes this message: Matrices.txt and Blank.txt have incomparable dimensions.

Summary of Runs

Input File	Threads	Result Notes
MatData.txt	2	All operations possible
MatData.txt	6	All operations possible
Matrices.txt	2	Some multiplications skipped (dimension mismatch)
Matrices.txt	12	Some multiplications skipped (dimension mismatch)

Matrices file

Input File (Matrices)

```
10,10
798576.17700638,335110.03257794,239261.77970727,180250.36056474,76102.43629483,784872.44692701,869727.93970759,919692.57770504,675206.70176422,835947.71489128
997049.7289848,847259.75446426,644659.48695373,474381.2891026,479268.99803732,435947.67225613,996928.44577671,693216.89125734,177913.56146235,627500.84654494
60984.09151221,340229.3080496,822385.22724443,881707.1135036,611034.87216013,378267.3040841,393895.77400803,475879.52667478,155256.59174094,184227.45893605
693983.81388528,867353.17906021,897538.66563503,316686.22068786,971781.20737011,575416.67063907,803121.13226303,121870.09370845,988037.3721695,203690.49667042
803085.73524438,748075.72602222,323792.28459791,363922.87558871,428327.36930566,797431.78135002,899520.04954761,854803.69558274,209371.48646589,332623.67331569
541156.44548425,90735.13226627,824331.65133871,718018.42938885,727993.26711252,827454.23287192,371968.17537775,245826.56254512,902493.29939029,229472.82155056
583866.16698572,277350.63088694,494605.85440039,98543.79548419,832294.30348471,302461.94099653,553164.65856759,465810.94717027,674469.36518262,202253.70036152
379636.48183964,498225.027994,546628.64933677,923498.36308126,954897.14146163,547989.61057067,918425.51250391,294755.17661438,411157.40144843,395707.42649072
763750.60477151,926618.55276521,580283.06716469,171206.88629496,132738.08968776,680082.26096899,155862.14443882,880478.94685777,510796.94456462,66029.01565315
113171.60173485,933387.68836858,356494.63453161,968223.82813175,476259.64010533,822070.06513313,65128.63587676,605528.71035777,264999.38557555,523235.38483168

10,10
697602.4905672,264999.38557555,523235.38483168,762410.08335423,774506.64866995,474755.06542305,494395.39228552,46517.69513601,340774.81657897,434313.44532762
298554.91249769,651322.15149353,300796.61062262,886799.00878278,926464.04709003,648702.57120778,730780.69292726,349331.47463962,526746.04852985,328836.6114068
72079.66818845,521507.67750129,529748.79964515,670358.10085482,407570.88303085,515772.1978345,447088.89968649,936116.76084245,263719.98981384,707046.70462684
453369.66172899,388008.74169257,485697.33832753,682589.70208863,369175.72664334,922866.08613741,455684.94402418,921650.4494319,379117.14146949,758411.48719835
79710.5849062,706731.73008264,291463.8221391,726451.05175992,942986.67557669,850944.6187115,723498.18924895,934638.93864959,545158.93147173,401801.10678966
320667.30276526,231011.3283608,160507.72699125,952530.70386208,508194.83819592,527830.43188614,481474.69863193,971004.44611481,698990.14295334,60254.52064602
134193.12776658,332388.94217422,392961.99851199,314028.54995055,971849.31008388,152209.9230793,655930.00321827,148713.69671342,626666.71321902,469201.33496123
498290.42668414,251774.12725837,640737.52756151,221991.32449839,321976.13701165,576841.77640405,885358.00246053,602528.31172032,347528.05859834,824551.76378444
198923.84938281,974070.33182956,631222.46119194,263113.17677927,494889.88263289,366156.52581208,596388.88840657,204021.37886027,244105.27813447,469201.33496123
469201.33496123,602528.31172032,599262.96677009,700676.91624391,665857.20314252,943573.31203124,480148.55402641,347528.05859834,608265.7328337,675187.78338983


12,10
599042.213,741597.841,51328.238,184811.701,935824.203,597004.400,22432.464,967101.469,98172.831,232239.960
974755.505,169100.212,614436.103,831624.641,134760.230,744391.138,744335.161,904949.852,19004.808,824133.620
589884.857,766152.189,133215.837,728087.288,325325.553,396915.032,257578.053,604001.417,519881.926,317992.722
507181.339,211466.660,530325.085,710438.626,524002.099,830754.061,867238.221,896528.691,142328.356,625963.873
127952.666,713376.682,455389.025,186805.938,614.681,72581.467,744724.839,299251.451,570598.100,849322.723
108552.359,371006.024,917934.657,594927.281,577765.546,886711.520,904789.874,189632.667,750885.301,208601.066
205119.062,251767.739,138532.712,627250.221,408610.544,999052.431,70737.579,286260.554,309217.816,745855.177
224968.418,336186.283,404121.440,54991.377,707804.920,474274.699,167769.595,447048.399,786812.674,595737.049
630172.872,281163.493,472548.367,73783.861,172199.831,712541.979,507344.676,422644.614,319752.184,6787.768
123016.251,645373.945,303667.337,572739.865,221982.224,999371.947,768377.571,971655.037,721431.561,902564.697
560796.922,467188.999,699669.348,500715.521,60834.846,494029.856,429671.666,355629.811,165579.750,39434.067
318543.520,225638.340,673074.506,53026.153,857828.762,100944.946,225355.647,799967.714,252238.919,718985.620

12,10
518109.206,165579.750,39434.067,278870.149,92376.620,954693.674,171960.459,124469.753,573524.445,303091.143
225638.340,673074.506,53026.153,857828.762,100944.946,225355.647,799967.714,252238.919,718985.620,718985.620
601375.369,997691.273,607380.040,975516.463,653155.209,922622.866,918044.577,290635.035,868282.477,318543.520
597355.854,998639.726,377477.647,859178.187,824721.394,736752.195,5863.527,902531.871,164113.158,743201.169
873358.862,828723.195,441413.925,765749.516,249431.459,724635.315,504705.554,413302.125,954191.035,769017.419
979251.673,453543.275,95195.968,963782.088,504214.627,789262.493,409870.107,584773.654,196392.260,943405.055
746839.137,58945.684,722533.917,632691.009,367090.332,727692.186,456532.203,613279.296,843405.055,739179.055
398551.407,815652.406,482430.589,353978.965,791879.667,790179.667,973983.674,964273.735,965723.644,920462.530
361891.623,657715.260,584773.654,490976.008,365067.425,582193.663,596788.613,845590.146,209501.358,974084.561
960734.927,104675.623,844302.117,404945.709,739567.996,724971.654,407238.404,102389.282,767344.446,354587.193
432509.470,917955.243,577022.407,878632.246,112204.846,677692.033,167749.607,652765.004,405546.593,451632.786
579557.674,125209.618,943405.055,682428.204,746839.137,537279.965,774179.779,121978.623,905642.714,846301.770
```

Compiling

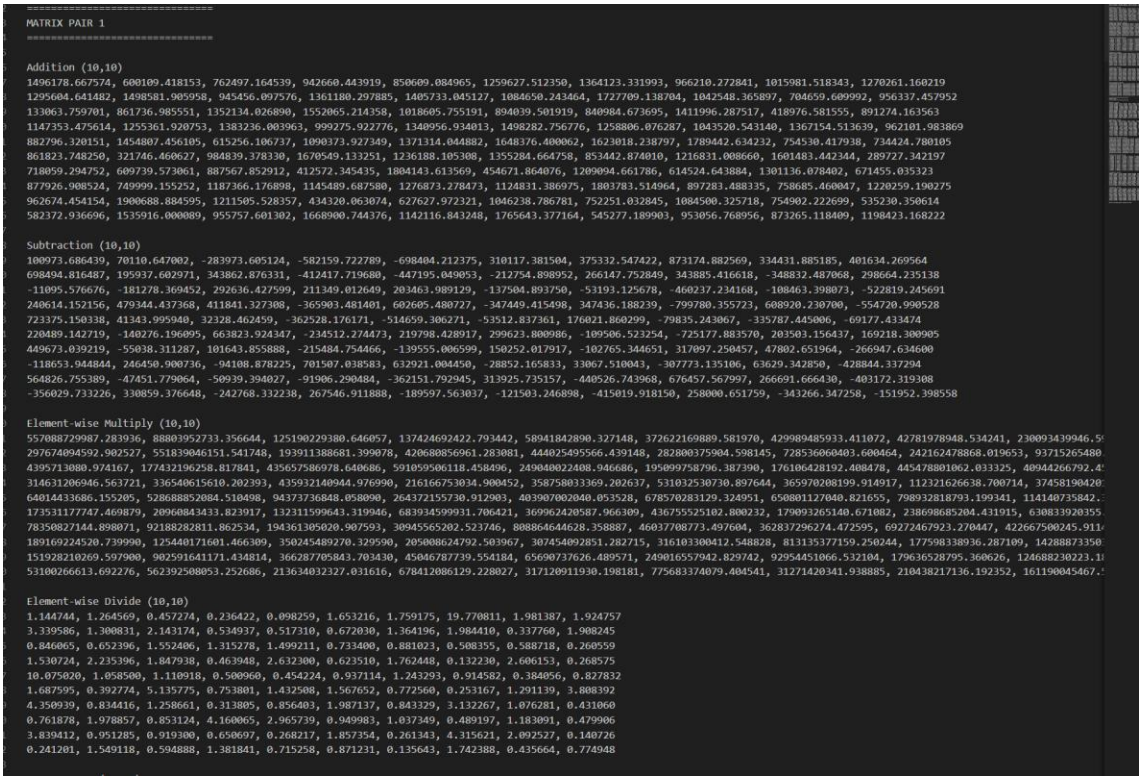
Martices.txt with 12 thread

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ ./task2 Matrices.txt 12
Matrix operations program starting...
Input file: Matrices.txt
Number of threads: 12
Output file: Task2_Matrices_12.txt

Processing matrix pair 1...
Processing matrix pair 2...

Processing completed. 2 matrix pairs processed.
Results written to Task2_Matrices_12.txt
```

Matrices.txt with 2 threads

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ gcc task2_advance.c -o Task2 -fopenmp -lm
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ ./task2 Matrices.txt 2
Matrix operations program starting...
Input file: Matrices.txt
Number of threads: 2
Output file: Task2_Matrices_12.txt

Processing matrix pair 1...
Processing matrix pair 2...

Processing completed. 2 matrix pairs processed.
Results written to Task2_Matrices_12.txt
```

OutputFile



```
==========
MATRIX PAIR 1
==========

Addition (10,10)
1496178.667574, 600109.418153, 762497.164539, 942660.443919, 850609.084965, 1259627.512350, 1364123.331993, 966210.272841, 1015981.518343, 1270261.160219
1295604.641482, 1498581.985958, 945456.097576, 1361180.297885, 1405733.045127, 1084650.243464, 1727709.138704, 1042548.365897, 704659.609992, 956337.457952
133063.759701, 861736.985551, 1352134.026890, 1552065.214358, 1018605.755191, 894039.501919, 840984.673695, 1411996.287517, 418976.581555, 891274.163563
1147353.475614, 1255361.920753, 1383236.003963, 999275.922776, 1340956.934013, 1498282.756776, 1258806.076287, 1043520.543140, 1367154.513639, 962101.983869
882796.320151, 1454807.456105, 615256.106737, 1090373.927349, 1371314.044882, 1648376.400062, 1623018.238797, 1789442.634232, 754530.417938, 734424.780105
861823.748250, 321746.460627, 984839.378330, 1670549.133251, 1236188.105308, 1355284.664758, 853442.874010, 1216831.008660, 1601483.442344, 289727.342197
718059.294752, 609739.573061, 887567.852912, 412572.345435, 1804143.613569, 454671.864076, 1209094.661786, 614524.643884, 1301136.078402, 671455.035323
877926.908524, 749999.155252, 1187366.176898, 1145489.687580, 1276873.278473, 1124831.386975, 1803783.514964, 897283.488335, 758685.460047, 1220259.190275
962674.454154, 1900688.884595, 1211505.528357, 434320.063074, 627627.972321, 1046238.786781, 752251.032845, 1084500.325718, 754902.222699, 535230.350614
582372.936696, 1535916.000089, 955757.601302, 1668900.744376, 1142116.843248, 1765643.377164, 545277.189903, 953056.768956, 873265.118409, 1198423.168222

Subtraction (10,10)
100973.686439, 70110.647002, -283973.605124, -582159.722789, -698404.212375, 310117.381504, 375332.547422, 873174.882569, 334431.885185, 401634.269564
698494.816487, 195937.602971, 343862.876331, -412417.719680, -447195.049053, -212754.898952, 266147.752849, 343885.416618, -348832.487068, 298664.235138
-11095.576676, -181278.369452, 292636.427599, 211349.012649, 203463.989129, -137504.893750, -53193.125678, -460237.234168, -108463.398073, -522819.245691
240614.152156, 479344.437368, 411841.327308, -365903.481401, 602605.480727, -347449.415498, 347436.188239, -799780.355723, 608920.230700, -554720.990528
723375.150338, 41343.995940, 32328.462459, -362528.176171, -514659.306271, -53512.837361, 176021.860299, -79835.243067, -335787.445006, -69177.433474
220489.142719, -140276.196095, 663823.924347, -234512.274473, 219798.428917, 299623.800986, -109506.523254, -725177.883570, 203503.156437, 169218.300905
449673.039219, -55038.311287, 101643.855888, -215484.754466, -139555.006599, 150252.017917, -102765.344651, 317097.250457, 47802.651964, -266947.634600
-118653.944844, 246450.900736, -94108.878225, 701507.038583, 632921.004450, -28852.165833, 33067.510043, -307773.135106, 63629.342850, -428844.337294
564826.755389, -47451.779064, -50939.394027, -91906.290484, -362151.792945, 313925.735157, -440526.743968, 676457.567997, 266691.666430, -403172.319308
-356029.733226, 330859.376648, -242768.332238, 267546.911888, -189597.563037, -121503.246898, -415019.918150, 258000.651759, -343266.347258, -151952.398558

Element-wise Multiply (10,10)
557088729987.283936, 88803952733.356644, 125190229380.646057, 137424692422.793442, 58941842890.327148, 372622169889.581970, 429989485933.411072, 42781978948.534241, 230093439946.5
297674094592.902527, 551839046151.541748, 193911388681.399078, 420680856961.283081, 444025495566.439148, 282800375904.598145, 728536060403.600464, 242162478868.019653, 93715265480
4395713080.974167, 177432196258.817841, 435657586978.640686, 591059506118.458496, 249048022408.946686, 195099758796.387390, 176106428192.408478, 445478801062.033325, 40944266792.4
314631206946.563721, 336540615610.202393, 435932140944.976990, 216166753034.900452, 358758033369.202637, 531032530730.897644, 365970208199.914917, 112321626638.700714, 37458190420
64014433686.155205, 528688852084.510498, 94373736848.058090, 264372155730.912903, 403907002040.053528, 678570283129.324951, 650801127040.821655, 798932818793.199341, 114140735842.
173531177747.469879, 20960843433.823917, 132311599643.319946, 683934599931.706421, 369962420587.966309, 436755525102.800232, 179093265140.671082, 238698685204.431915, 630833920355.
78350827144.898071, 92188282811.862534, 194361305020.907593, 30945565202.523746, 808864644628.358887, 46037708773.497604, 362837296274.472595, 69272467923.270447, 422667500245.911
189169224520.739990, 125440171601.466309, 350245489270.320590, 205008624792.503967, 30745409285l.282715, 316103300412.548828, 813135377159.250244, 177598338936.287109, 14288873350
151928210269.597900, 9025916411l.434814, 366287705843.703430, 450467877l9.554184, 65690737626.489571, 249016557942.829742, 92954451066.532104, 179636528795.360626, 124688230223.1
53100266613.692276, 562392508053.252686, 213634032327.031616, 678412086129.228027, 317120911930.198181, 77568337407l.404541, 3127142034l.938885, 210438217136.192352, 161190045467.

Element-wise Divide (10,10)
1.144744, 1.264569, 0.457274, 0.236422, 0.098259, 1.653216, 1.759175, 19.770811, 1.981387, 1.924757
3.339586, 1.300831, 2.143174, 0.534937, 0.517310, 0.672030, 1.364196, 1.984410, 0.337760, 1.908245
0.846065, 0.652396, 1.552406, 1.315278, 1.499211, 0.733400, 0.881023, 0.508355, 0.588718, 0.260559
1.530724, 2.235396, 1.847938, 0.463948, 2.632300, 0.623510, 1.762448, 0.132230, 2.606153, 0.268575
10.075020, 1.058500, 1.110918, 0.500960, 0.454224, 0.937114, 1.243293, 0.914582, 0.384056, 0.827832
1.687595, 0.392774, 5.135775, 0.753801, 1.432508, 1.567652, 0.772560, 0.253167, 1.291139, 3.808392
4.350939, 0.834416, 1.258661, 0.313805, 0.856403, 1.987137, 0.843329, 3.132267, 1.076281, 0.431060
0.761878, 1.978857, 0.853124, 4.160065, 2.965739, 0.949983, 1.037349, 0.489197, 1.183091, 0.479906
3.839412, 0.951285, 0.919300, 0.650697, 0.268217, 1.857354, 0.261343, 4.315621, 2.092527, 0.140726
0.241201, 1.549118, 0.594888, 1.381841, 0.715258, 0.871231, 0.135643, 1.742388, 0.435664, 0.774948
```

```
Element-wise Multiply (12,10)
310369285337.912903, 122793585113.319748, 2024081176.283946, 51538466594.813446, 86448276787.333847, 569956324030.165649, 3857496805.940976, 120374880972.367157, 56304518413.35379
2199422140 54.061707, 113817041656.395279, 32581182806.401756, 713391536237.724365, 13603364140.297581, 167752746525.056305, 595444097194.991943, 228263572417.689972, 13664183662.8
354742223545.887207, 764383352755.146606, 80912640405.693481, 710261135945.022339, 212488079562.755585, 366202884382.321716, 236468134710.868622, 175543972969.844574, 45140436645
302967741891.208496, 211179007400.535156, 200185865230.874969, 610393370661.451172, 432155741546.205994, 612059877946.913818, 5085074724.265467, 809145716893.410889, 23357955976.1
111748594767.626083, 591191803145.538940, 201015056927.173126, 143046556609.425995, 153320778.649579, 52595094202.707108, 375866762445.055847, 123681260607.633377, 544459591608.03
106300079158.846603, 168267287169.688599, 87383678233.862976, 573380257090.342651, 291317839269.841309, 699848144847.019409, 370846322468.896545, 110892187599.355209, 14746806126 4.
153190943246.329498, 14840621584.488476, 100094583033.992905, 396855575219.962952, 149996980255.660614, 727002647443.004150, 32293982775.756535, 175557671029.689972, 260795869110.
89661479524.464127, 274211150593.146881, 194960544326.728149, 194657907 14.384808, 560496324350.561646, 374762223722.345276, 163404846523.592041, 431077029429.500244, 759843602680.
228054283418.651245, 184925519901.003204, 276333835262.322998, 36226105528.606888, 62864548888.605179, 414837424795.279053, 302777525502.974365, 357384120858.373596, 66988516771.4
118186008924.298691, 67554919760.842735, 256386975492.852417, 231928550704.989258, 164170948551.303101, 724516333377.790405, 312912855683.436646, 99487061590.113449, 553586501502.
242549979511.851318, 428858591103.971802, 403724891287.080627, 439944802823.290222, 6825964526.863716, 334800097475.337280, 72077253110.535263, 232142694999.934235, 67150303482.29
184614341518.972504, 28252090357.554119, 634981891352.027954, 36186542356.819214, 640660092305.858398, 54235697053.806885, 174465784990.862000, 97578960198.177826, 228438339179.58

Element-wise Divide (12,10)
1.156208, 4.478796, 1.301622, 0.662716, 10.130531, 0.625336, 0.130451, 7.769771, 0.171175, 0.766238
4.319991, 0.251236, 11.587416, 0.969453, 1.334987, 3.303184, 0.930457, 3.587669, 0.026433, 1.146245
0.980893, 0.767925, 0.219329, 0.746361, 0.498083, 0.430203, 0.280572, 2.078213, 0.598747, 0.998271
0.849044, 0.211755, 1.404918, 0.826882, 0.635369, 1.127590, 147.903851, 0.993349, 0.867257, 0.842254
0.146506, 0.860814, 1.031660, 0.243952, 0.002464, 0.100163, 1.475563, 0.724050, 0.597991, 1.104426
0.110852, 0.818017, 9.642579, 0.617284, 1.145872, 1.123468, 2.207504, 0.324284, 3.823396, 0.221115
0.274650, 4.271182, 0.191732, 0.991401, 1.113106, 1.372905, 0.154945, 0.466770, 0.366630, 1.009032
0.564465, 0.412169, 0.837678, 0.155352, 0.893829, 0.600211, 0.172251, 0.463612, 0.814739, 0.647215
1.741330, 0.427485, 0.808088, 0.150280, 0.471693, 1.223892, 0.850125, 0.499822, 1.526254, 0.006968
0.128044, 6.165466, 0.359667, 1.414362, 0.300151, 1.378498, 1.886800, 9.489812, 0.940167, 2.545396
1.296612, 0.508945, 1.212551, 0.569881, 0.542177, 0.728989, 2.561387, 0.544805, 0.408288, 0.087314
0.549632, 1.802085, 0.713452, 0.077702, 1.148612, 0.187881, 0.291090, 6.558262, 0.278519, 0.849562

Transpose A (10,12)
599042.213000, 974755.505000, 589884.857000, 507181.339000, 127952.666000, 108552.359000, 205119.062000, 224968.418000, 630172.872000, 123016.251000, 560796.922000, 318543.520000
741597.841000, 169100.212000, 766152.189000, 211466.660000, 713376.682000, 371006.024000, 251767.739000, 336186.283000, 281163.493000, 645373.945000, 467188.999000, 225638.340000
51328.238000, 614436.103000, 133215.837000, 530325.085000, 455389.025000, 917934.657000, 138532.712000, 404121.440000, 472548.367000, 303667.337000, 699669.348000, 673074.506000
184811.701000, 831624.641000, 728087.288000, 710438.626000, 186805.938000, 594927.281000, 627250.221000, 54991.377000, 73783.861000, 572739.865000, 500715.521000, 53026.153000
935824.203000, 134760.230000, 325325.553000, 524002.099000, 614.681000, 577765.546000, 408610.544000, 707804.920000, 172199.831000, 221982.224000, 60834.846000, 857828.762000
597004.400000, 744391.138000, 396915.032000, 830754.061000, 72581.467000, 886711.520000, 999052.431000, 474274.699000, 712541.979000, 999371.947000, 494029.856000, 100944.946000
22432.464000, 744335.161000, 257578.053000, 867238.221000, 744724.839000, 904789.874000, 70737.579000, 167769.595000, 507344.676000, 768377.571000, 429671.666000, 225355.647000
967101.469000, 904949.852000, 604001.417000, 896528.691000, 299251.451000, 189632.667000, 286260.554000, 447048.399000, 422644.614000, 971655.037000, 355629.811000, 799967.714000
98172.831000, 19004.808000, 519881.926000, 142328.356000, 570598.100000, 750885.301000, 309217.816000, 786812.674000, 319752.184000, 721431.561000, 165579.750000, 252238.919000
232239.960000, 824133.620000, 317992.722000, 625963.873000, 849322.723000, 208601.066000, 745855.177000, 595737.049000, 6787.768000, 902564.697000, 39434.067000, 718985.620000

Transpose B (10,12)
518109.206000, 225638.340000, 601375.369000, 597355.854000, 873358.862000, 979251.673000, 746839.137000, 398551.407000, 361891.623000, 960734.927000, 432509.470000, 579557.674000
165579.750000, 673074.506000, 997691.273000, 998639.726000, 828723.195000, 453543.275000, 58945.684000, 815652.406000, 657715.260000, 104675.623000, 917955.243000, 125209.618000
39434.067000, 53026.153000, 607380.040000, 377477.647000, 441413.925000, 95195.968000, 722533.917000, 482430.589000, 584773.654000, 844302.117000, 577022.407000, 943405.055000
278870.149000, 857828.762000, 975516.463000, 859178.187000, 765749.516000, 963782.088000, 632691.009000, 353978.965000, 490976.008000, 404945.709000, 878632.246000, 682428.204000
92376.620000, 100944.946000, 653155.209000, 824721.394000, 249431.459000, 504214.627000, 367090.332000, 791879.667000, 365067.425000, 739567.996000, 112204.846000, 746839.137000
954693.674000, 225355.647000, 922622.866000, 736752.195000, 724635.315000, 789262.493000, 727692.186000, 790179.667000, 582193.663000, 724971.654000, 677692.033000, 537279.965000
171960.459000, 799967.714000, 918044.577000, 5863.527000, 504705.554000, 409870.107000, 456532.203000, 973983.674000, 596788.613000, 407238.404000, 167749.607000, 774179.779000
124469.753000, 252238.919000, 290635.035000, 902531.871000, 413302.125000, 584773.654000, 613279.296000, 964273.735000, 845590.146000, 102389.282000, 652765.004000, 121978.623000
573524.445000, 718985.620000, 868282.477000, 164113.158000, 954191.035000, 196392.260000, 843405.055000, 965723.644000, 209501.358000, 767344.446000, 405546.593000, 905642.714000
303091.143000, 718985.620000, 318543.520000, 743201.169000, 769017.419000, 943405.055000, 739179.055000, 920462.530000, 974084.561000, 354587.193000, 451632.786000, 846301.770000

Matrix multiplication cannot be done
```

MatData file

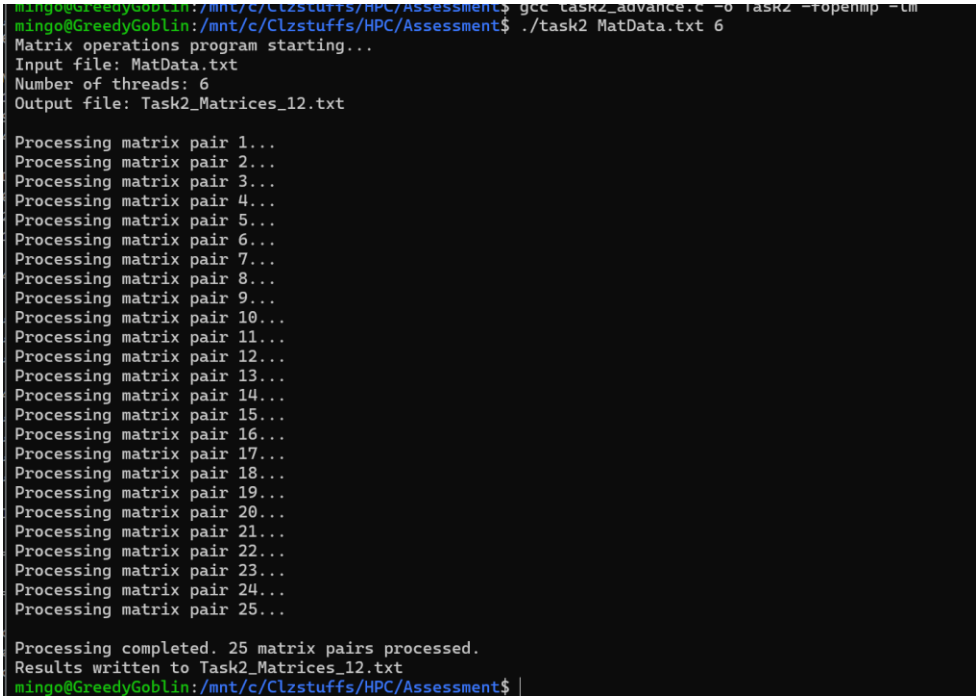
Input File (MatData)

```
3,4
96087.460425,693590.143567,372212.092347,152859.228670
621349.540618,324558.949061,384380.257759,995051.046325
264321.856225,837845.185678,847886.766896,49788.599073

3,4
807623.015898,314866.253882,929529.038283,191716.041160
359139.693323,317625.201988,833649.519395,324261.752064
617795.138471,594120.097081,585594.901251,726643.288179

4,6
829878.482878,128053.107625,933444.808093,195740.374456,42462.518964,395468.811322
793478.381145,269370.848737,998874.407761,445817.468455,395833.447548,343773.710811
113235.400602,831114.360170,418088.724234,401083.191627,13430.104739,624007.543713
714549.069657,525986.836701,81015.748839,777302.827031,103710.903016,272526.828896

6,2
930489.946963,382161.961067
579206.498398,496647.994188
711238.873273,251224.622005
28484.704706,70524.870217
93657.711183,487315.809759
985084.235032,215947.439224

5,5
544361.163794,54829.061045,853064.818198,335026.212154,545710.684365
507587.777340,702915.673301,75591.247272,123290.815995,125645.488386
346026.731019,652582.213908,261269.676728,519032.243669,61813.329999
112129.302028,493964.084891,888502.000422,87019.479496,44328.795553
844964.837321,733879.064327,76787.516152,616035.342010,481306.166399

5,5
122395.281813,648662.572214,957713.566965,305220.177753,212447.336688
719223.615732,828614.807904,86132.147965,665569.656715,33998.969848
171024.743308,632451.780528,441091.020498,880941.043458,156543.705361
311981.081983,76574.310165,96872.856674,87023.502999,806767.110613
161849.815991,210322.487244,73918.314436,621114.938369,347715.561420

8,4
283787.059742,393203.844784,845554.562370,647635.789551
402888.308356,567769.487747,128644.615482,667428.589607
842184.023963,23326.058143,658458.159698,473045.266451
244202.729792,789196.653194,796158.253192,320251.578878
179152.880305,887232.704466,544837.318512,429445.291698
207577.601883,959221.857715,136934.370797,591528.155214
301503.384297,709552.533614,110626.061431,307943.298821
375266.486763,976894.273788,126615.849525,814253.633346

4,8
244742.183129,354997.337619,32350.989135,806292.771451,989992.705322,604570.953930,321357.946121,983975.669835
188905.288339,798634.811429,998621.529523,459887.077514,595754.443027,638633.742936,33536.700944,126700.720934
585265.361379,953523.162582,247434.644402,475202.930279,352302.468884,300331.560384,65422.228950,68706.134707
281310.216104,625582.758416,790005.587966,106166.059239,345441.856266,488856.730169,417707.491837,628663.624790

6,6
819196.375487,288450.480804,494467.281202,636539.873673,829742.103733,860859.767029
```

MataData with 2 threads

```
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ gcc task2_advance.c -o Task2 -fopenmp -lm
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$ ./task2 MatData.txt 2
Matrix operations program starting...
Input file: MatData.txt
Number of threads: 2
Output file: Task2_Matrices_12.txt

Processing matrix pair 1...
Processing matrix pair 2...
Processing matrix pair 3...
Processing matrix pair 4...
Processing matrix pair 5...
Processing matrix pair 6...
Processing matrix pair 7...
Processing matrix pair 8...
Processing matrix pair 9...
Processing matrix pair 10...
Processing matrix pair 11...
Processing matrix pair 12...
Processing matrix pair 13...
Processing matrix pair 14...
Processing matrix pair 15...
Processing matrix pair 16...
Processing matrix pair 17...
Processing matrix pair 18...
Processing matrix pair 19...
Processing matrix pair 20...
Processing matrix pair 21...
Processing matrix pair 22...
Processing matrix pair 23...
Processing matrix pair 24...
Processing matrix pair 25...

Processing completed. 25 matrix pairs processed.
Results written to Task2_Matrices_12.txt
mingo@GreedyGoblin:/mnt/c/Clzstuffs/HPC/Assessment$
```

MatData with 6 threads



Output File

```
==============================
MATRIX PAIR 1
==============================

Addition (3,4)
903710.476323, 1008456.397449, 1301741.130630, 344575.269830
980489.233941, 642184.151049, 1218029.777154, 1319312.798389
882116.994696, 1431965.282759, 1433481.668147, 776431.887252

Subtraction (3,4)
-711535.555473, 378723.889685, -557316.945936, -38856.812490
262209.847295, 6933.747073, -449269.261636, 670789.294261
-353473.282246, 243725.088597, 262291.865645, -676854.689106

Element-wise Multiply (3,4)
77602444578.418213, 218388130234.419861, 345981948236.610107, 29305566175.383572
223151283463.935425, 103088101752.513138, 320438417145.716553, 322656995674.460938
163296757767.435638, 497780663053.861816, 496518167532.492798, 36178551344.230629

Element-wise Divide (3,4)
0.118976, 2.202809, 0.400431, 0.797321
1.730105, 1.021830, 0.461081, 3.068666
0.427847, 1.410229, 1.447907, 0.068519

Transpose A (4,3)
96087.460425, 621349.540618, 264321.856225
693590.143567, 324558.949061, 837845.185678
372212.092347, 384380.257759, 847886.766896
152859.228670, 995051.046325, 49788.599073

Transpose B (4,3)
807623.015898, 359139.693323, 617795.138471
314866.253882, 317625.201988, 594120.097081
929529.038283, 833649.519395, 585594.901251
191716.041160, 324261.752064, 726643.288179

Matrix multiplication cannot be done

==============================
MATRIX PAIR 2
==============================

Addition cannot be done (different size)
Subtraction cannot be done (different size)
Element-wise operations cannot be done (different size)

Transpose A (6,4)
829878.482878, 793478.381145, 113235.400602, 714549.069657
128053.107625, 269370.848737, 831114.360170, 525986.836701
933444.808093, 998874.407761, 418088.724234, 81015.748839
195740.374456, 445817.468455, 401083.191627, 777302.827031
42462.518964, 395833.447548, 13430.104739, 103710.903016
395468.811322, 343773.710811, 624007.543713, 272526.828896

Transpose B (2,6)
930489.946963, 579206.498398, 711238.873273, 28484.704706, 93657.711183, 985084.235032
382161.961067, 496647.994188, 251224.622005, 70524.870217, 487315.809759, 215947.439224

Matrix Multiply (4,2)
```

```
==============================
MATRIX PAIR 25
==============================

Addition (10,10)
1571441.716082, 752417.484059, 1488280.635704, 1908125.613453, 805193.488455, 1577113.787612, 741031.618992, 1339451.095127, 919339.461496, 1045332.810084
715181.053148, 981137.206326, 1579471.551152, 1324128.999302, 385829.733727, 1702985.834752, 1544363.287663, 437767.013986, 1352803.878588, 1096269.028253
1279654.953776, 1685231.426335, 708067.244412, 830252.568989, 103998.389881, 1663646.119344, 739422.155945, 796998.354407, 1423766.510018, 1150250.532828
660785.989313, 238609.706227, 1471146.750692, 1527770.860062, 547578.465004, 1202377.756156, 194343.359856, 1613832.355021, 1498032.887786, 485267.713921
371411.778435, 912473.793850, 1518774.530107, 1493322.087177, 483696.559254, 1116215.288504, 943334.957922, 273908.518201, 703333.102111, 1146945.365454
1371171.326855, 1729293.142861, 338512.052269, 1192297.470173, 138840.342579, 1009513.382607, 1441244.962663, 1442664.666738, 261414.994521, 1133763.013412
1315737.933753, 679438.974057, 753756.221254, 1587501.419687, 955391.498524, 1587222.663587, 84222.899726, 984724.809919, 841470.336966, 616232.117190
558437.714695, 443099.678679, 973683.691777, 706235.261314, 1166968.932887, 979612.368530, 226276.761609, 719384.497381, 1082270.224771, 620741.282384
442553.653494, 1291249.108563, 993161.106862, 136493.326488, 1085420.003031, 901941.890844, 696402.820776, 602902.562797, 1663427.381285, 987693.756556
1594566.375486, 1308787.578691, 1476875.513100, 1343809.839028, 832572.756972, 1093763.735641, 1221027.001865, 1218838.402735, 1546922.289050, 1187675.953011

Subtraction (10,10)
-351393.077348, -547796.996627, 392821.135114, 55975.116207, 73185.073183, 110691.597654, 228093.652440, -463075.102055, -73374.636126, -133166.167210
104517.306268, -87430.327998, -140727.651532, 244122.407268, -175551.335079, -207531.102410, 25390.687543, -281195.009638, 42433.676986, -902742.708727
-396195.938636, 66646.327549, -566958.796116, -351327.890177, -8842.580125, 183214.420004, 566577.770689, 314655.714389, 493708.559358, 241862.281790
133569.539287, -123034.525675, 475949.276078, 294925.538194, -196519.913984, -536406.122094, 105866.688306, 80311.397109, 221103.613664, -210904.583463
-13473.923455, 748473.462796, -245942.312267, -355303.671829, 34395.889058, 399958.307096, -375489.909218, 47670.516531, 683295.388875, 2059.977786
542525.323185, -201664.113919, -32554.424001, 462853.027009, 133151.992309, -479971.793373, -412817.908373, -475846.953290, 119093.327053, -758082.075076
650832.106653, -10265.793763, -284244.961436, -34179.290425, -539875.189306, 348678.719005, -68831.279216, -451974.827111, -564858.014870, -125773.598134
5896.724715, -196778.389397, -269325.310655, 121898.376278, 523708.177539, 845468.030366, -34719.214175, -270540.494535, 321129.860381, -8736.530656
407609.378006, -139877.174039, -493557.493236, 38187.705092, -601954.389037, -647662.794224, -343334.591964, -474086.612393, 81598.587569, 292075.313246
-330651.702576, -69439.051073, -133482.172740, -415167.384068, 763556.085906, 734086.650727, 328042.577609, 448167.393839, 145894.372354, -645729.062205

Element-wise Multiply (10,10)
586487993058.660889, 66512630201.028519, 515167701604.812744, 909452535770.250122, 160745124728.382782, 618758817321.168457, 124275286515.621292, 394922671523.417725, 209950
125140017868.090866, 238746538845.837463, 618731527248.198242, 423430464265.548157, 29511578044.957100, 714272898724.614014, 596103319316.840942, 28142331272.225952, 4570694
370136394732.948914, 708890806832.832397, 44979236528.974724, 141472010474.153015, 2684368468.643360, 683537721677.737183, 56433688615.858704, 134049539582.453506, 445840733
104699325461.754684, 10449274349.415754, 484436262168.114319, 561775681943.890625, 65305524685.964523, 28949185169.703552, 6640396456.809739, 649501237401.756836, 548803931
34441290636.744583, 68098974988.294281, 561547113084.625000, 525942539208.875366, 58194821062.517029, 271492480718.742401, 187222042728.289886, 18188349549.183800, 694621601
396444270323.078125, 737446589775.834839, 28382654752.333321, 301835083192.385437, 386796917.899521, 197186086807.233337, 476692104232.003418, 463712754424.722351, 135386447
326895969816.557983, 115382983236.511047, 121838310744.338974, 629748133403.571045, 155326923855.936218, 599424733679.310791, 588937959.936224, 191350426731.998077, 97251937
77954477457.852417, 39403897677.938919, 218880952168.268921, 120977257546.010254, 271886558775.550690, 61206050551.498253, 12498937252.831675, 111080473972.217911, 267046113
7427082795.612306, 411939659136.901367, 185692496263.487183, 4293031838.891550, 203946874124.727051, 98508019859.302673, 91774561686.420807, 34683346044.229980, 690083080828
608327844353.537842, 427025786080.482971, 540835947688.749268, 408365231668.648254, 27539874831.966530, 164358974656.942719, 345823751639.768494, 321178259770.291199, 592920

Element-wise Divide (10,10)
```