Systems Programming HW5 Report
Mohammad Ashraf Yawar 161044123

- HOW TO RUN AND TEST THE PROGRAM ?

- You can find the instructions in README.txt in order to run and test the program.

```
-HOW TO RUN THE PROGRAM:
> Run below commands in order:

alias vg='valgrind --leak-check=full -v --track-origins=yes --log-file=vg_logfile.out'

make

*****************************************************************************************************************************************************

- HOW TO TEST THE PROGRAM:

vg ./hw5 -i inputFile1.txt -j inputFile2.txt -o outputFile.csv -n 4 -m 4

*****************************************************************************************************************************************************
```

Implemented Concepts:

- File read, write,System-Calls.
- Signal handling, threads, join-able threads.
- Make files.
- Waiting for the threads to finish.
- Mutex, condition variables, synchronization barrier.

Working Cases:

- This program works for cases all the cases.

Note Working Cases:

- NONE

Design Explanation:

- All the System-Calls and their possible return error values are checked with detailed errno checks.
- I have my global and constant variable and constants as:

```c
#define MESSAGE_LEN 1024
#define BUFFER_LIMIT 1// buffer size used to read from file.
#define ACCESS_PERMISSION_FLAG_INPUT  O_RDONLY | O_CREAT
#define MODE S_IRUSR | S_IWUSR | S_IXUSR | S_IRGRP | S_IWGRP | S_IXGRP | S_IROTH | S_IWOTH | S_IXOTH
#define PI 3.141

sig_atomic_t sigintcaught = 0;

pthread_t *threadsArray=NULL;
pthread_mutex_t mtx;
pthread_cond_t cond;
int arrived = 0,M = 0,N;
uint64_t **matrixA = NULL,**matrixB = NULL,**matrixC = NULL;
struct indexKeeper{
    int actualIndex;
    int startingIndex;
};
struct dftStruct{
    float realNum;
    float imgNum;
};
struct indexKeeper *indKeeperArr = NULL;
struct dftStruct **dft2dArr = NULL;
```

- In my main program I start the program by controlling and getting some inputs from the argv pointers as:

```c
int main(int argc, char **argv){
    setbuf(stdout, NULL);
    char *input_path = NULL,*message;
    int inpfd = 0,s;
    mode_t mode = MODE;
    pthread_t supplierThreadPointer;
    pthread_attr_t attr;
    struct sigaction newact;
    time_t t;time(&t);

    newact.sa_handler = &siginthandler; /* set the new handler */
    newact.sa_flags = 0;
    if ((sigemptyset(&newact.sa_mask) == -1) || (sigaction(SIGINT,&newact, NULL) == -1)){
        perror("Failed to install SIGINT signal handler");
        exit(EXIT_FAILURE);
    }

    if (sigintcaught == 1){// if sigint has recieved.
        exit(EXIT_FAILURE);
    }

    // check if the user has entered sufficient arguments.
    if (argc < 7){
        perror("No Sufficient Parameters !!!\n");
        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Usage: vg ./hw4 -C 10 -N 5 -F inputfilePath\n",ctime(&t));
        printMessage(message);
        exit(EXIT_FAILURE);
    }else if (argc > 7){
        perror("Too Much Parameters !!!\n");
        message = (char*) malloc(MESSAGE_LEN * sizeof(char));
        sprintf(message,"[%.19s] Usage: vg ./hw4 -C 10 -N 5 -F inputfilePath\n",ctime(&t));
        printMessage(message);
        exit(EXIT_FAILURE);
    }

    //file paths read from terminal
    C = atoi(argv[2]);
    N = atoi(argv[4]);
    input_path = argv[6];// input file path
    if (C < 4 && N < 1){
        perror("error:");
        exit(EXIT_FAILURE);
    }
    //input file settings:
    inpfd = open(input_path,ACCESS_PERMISSION_FLAG_INPUT,mode);
    if (inpfd == -1){// if file not found then print error on sdterr
        perror("Coudln't Open The Input File !!!\n");
        exit(EXIT_FAILURE);
    }
    //*********************************************************************************************
```

- In my main thread I first allocate spaces for all the matrix pointers, initialize the mutex and condition variable, create the m threads and print the result, all in order.

```
    // initialize and fill in MatrixB
    indCounter = 0;
    matrixB = (uint64_t**) calloc(matrixHalfSize, sizeof(uint64_t*));
    for (int i = 0; i < matrixHalfSize; ++i){
        matrixB[i] = (uint64_t* ) calloc(matrixHalfSize,sizeof(uint64_t));
    }
    for (int i = 0; i < matrixHalfSize; ++i){
        for (int j = 0; j < matrixHalfSize; ++j){
            matrixB[i][j] = buf1[indCounter];
            ++indCounter;
        }
    }

    // initialize MatrixC
    matrixC = (uint64_t**) calloc(matrixHalfSize, sizeof(uint64_t*));
    for (int i = 0; i < matrixHalfSize; ++i){
        matrixC[i] = (uint64_t* ) calloc(matrixHalfSize,sizeof(uint64_t));
    }

    // initialize dft2dArr
    dft2dArr = (struct dftStruct**) calloc(matrixHalfSize, sizeof(struct dftStruct*));
    for (int i = 0; i < matrixHalfSize; ++i){
        dft2dArr[i] = (struct dftStruct* ) calloc(matrixHalfSize,sizeof(struct dftStruct));
    }

    // initialize indKeeperArr
    indKeeperArr = (struct indexKeeper*) calloc(matrixHalfSize,sizeof(struct indexKeeper));
    //*****************************************************************************************
```

```
// mutex initialization settings:
s = pthread_mutexattr_init(&mtxAttr);
if (s != 0){
    perror("pthread_mutexattr_init");
    close(inpfd1);close(inpfd2);close(outputFd);
    freeMatrixes();
    free(indKeeperArr);
    exit(EXIT_FAILURE);
}
s = pthread_mutexattr_settype(&mtxAttr, PTHREAD_MUTEX_ERRORCHECK);
if (s != 0){
    perror("pthread_mutexattr_settype");
    close(inpfd1);close(inpfd2);close(outputFd);
    freeMatrixes();
    free(indKeeperArr);
    exit(EXIT_FAILURE);
}
s = pthread_mutex_init(&mtx, &mtxAttr);
if (s != 0){
    perror("pthread_mutex_init");
    close(inpfd1);close(inpfd2);close(outputFd);
    freeMatrixes();
    free(indKeeperArr);
    exit(EXIT_FAILURE);
}
s = pthread_mutexattr_destroy(&mtxAttr);
if (s != 0){
    perror("pthread_mutexattr_destroy");
    close(inpfd1);close(inpfd2);close(outputFd);
    freeMatrixes();
    free(indKeeperArr);
    exit(EXIT_FAILURE);
}
//*****************************************************************************************

// condition variable initialization settings:
s = pthread_cond_init(&cond, NULL);
if (s != 0){
    perror("pthread_cond_init");
    close(inpfd1);close(inpfd2);close(outputFd);
    freeMatrixes();
    free(indKeeperArr);
    exit(EXIT_FAILURE);
}
//*****************************************************************************************
```

- Later I wait for the threads to terminate and exit the program:

```
// wait for the threads to terminate:
for (int i = 0; i < M; ++i){
    s = pthread_join(threadsPointersList[i],NULL);
    if (s != 0) {
        perror("pthread_join() error");
        close(inpfd1);close(inpfd2);close(outputFd);
        freeThreadsPointersArr();
        freeMatrixes();
        free(indKeeperArr);
        exit(EXIT_FAILURE);
    }
}
//*****************************************************************************************
```

-Inside myThread funtion:

- I find the starting index of the matrix for multiplication and calcualte the matrix C for this particular thread's share.

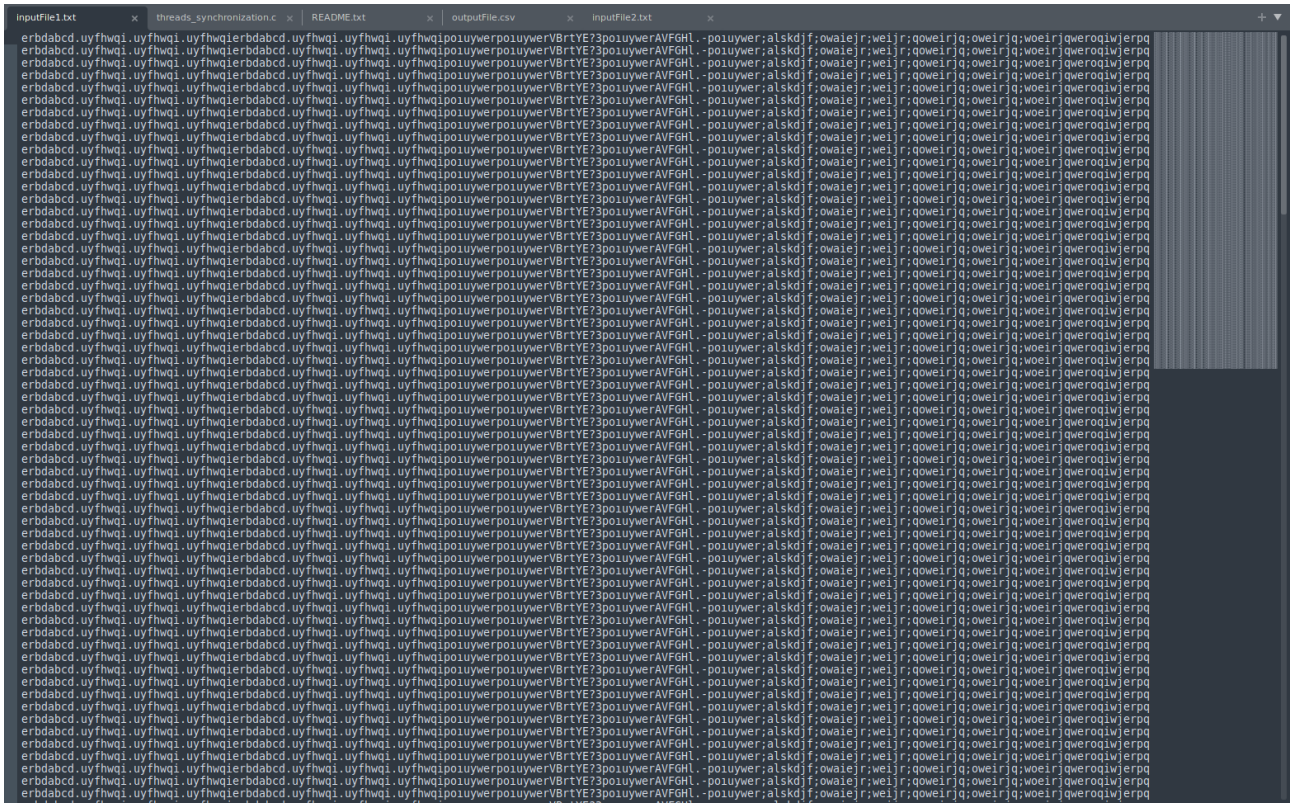
- After than I set up the barrier using threads mutex and condition variable, so that all the threads will calculate the first task and then advance to the second part.

-Later threads will calculate the second task and return gracefully.

```
void * myThread(void* ptr){
    int ptrLocal = *((int *) ptr),startingInd = 0,gapeNum = power(2,N)/M;
    char* message;
    time_t t;time(&t);
    int size = power(2,N);
    clock_t tt;tt = clock();
    double time_taken;

    // find the starting index of the matrix for multiplification
    for (int i = 0; i < M; ++i){
        if (indKeeperArr[i].actualIndex == ptrLocal){
            startingInd = indKeeperArr[i].startingIndex;
        }
    }

    // calcualte the matrix C for this partecular thread's share
    for (int i = startingInd; i < startingInd + gapeNum; i++) {
        for (int j = 0; j < size; j++) {
            matrixC[i][j] = 0;
            for (int k = 0; k < size; k++){
                matrixC[i][j] += matrixA[i][k] * matrixB[k][j];
            }
        }
    }

    tt = clock() - tt;
    time_taken = ((double)tt)/CLOCKS_PER_SEC;
    message = (char*) malloc(MESSAGE_LEN * sizeof(char));
    sprintf(message,"[%.19s] Thread %d has reached the rendezvous point in %f seconds.\n",ctime(&t),ptrLocal,time_taken);
    printMessage(message);

    // barrier part START
    pthread_mutex_lock(&mtx);
    ++arrived;
    while(arrived < M){pthread_cond_wait(&cond,&mtx);}
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mtx);
    // barrier part END

    tt = clock();
    message = (char*) malloc(MESSAGE_LEN * sizeof(char));
    sprintf(message,"[%.19s] Thread %d is advancing to the second part\n",ctime(&t),ptrLocal);
    printMessage(message);
```

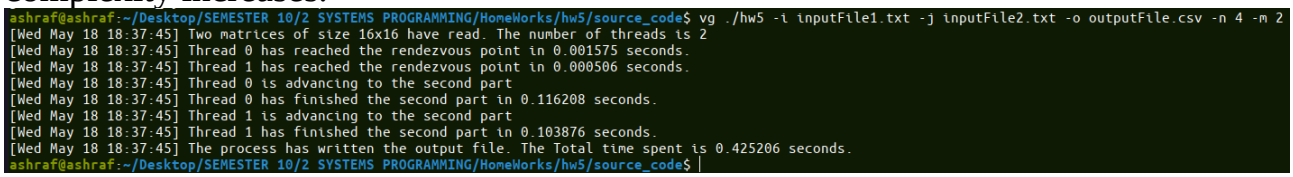
```
    //calcualte the 2D DFT
    int c = 0;
    for(int i=startingInd;i<startingInd+gapeNum;i++){

        for(int j=0;j<size;j++){

            float ak=0,bk=0;

            for(int ii=0;ii<size;ii++){

                for(int jj=0;jj<size;jj++){

                    float x = -2.0*M_PI*i*ii/(float)size;
                    float y =- 2.0*M_PI*j*jj/(float)size;
                    ak += matrixC[ii][jj]*cos(x+y);
                    bk += matrixC[ii][jj]*1.0*sin(x+y);
                }
            }

            dft2dArr[i][j].realNum = ak;
            dft2dArr[i][j].imgNum = bk;

            temp[c][j].realNum = ak;
            temp[c][j].imgNum = bk;
        }
        ++c;
    }
    if (sigintcaught == 1){// if sigint has recieved.
        free(ptr);
        pthread_exit(temp);
    }

    tt = clock() - tt;
    time_taken = ((double)tt)/CLOCKS_PER_SEC;
    message = (char*) malloc(MESSAGE_LEN * sizeof(char));
    sprintf(message,"[%.19s] Thread %d has finished the second part in %f seconds.\n",ctime(&t),ptrLocal,time_taken);
    printMessage(message);

    free(ptr);
    pthread_exit(temp);
```

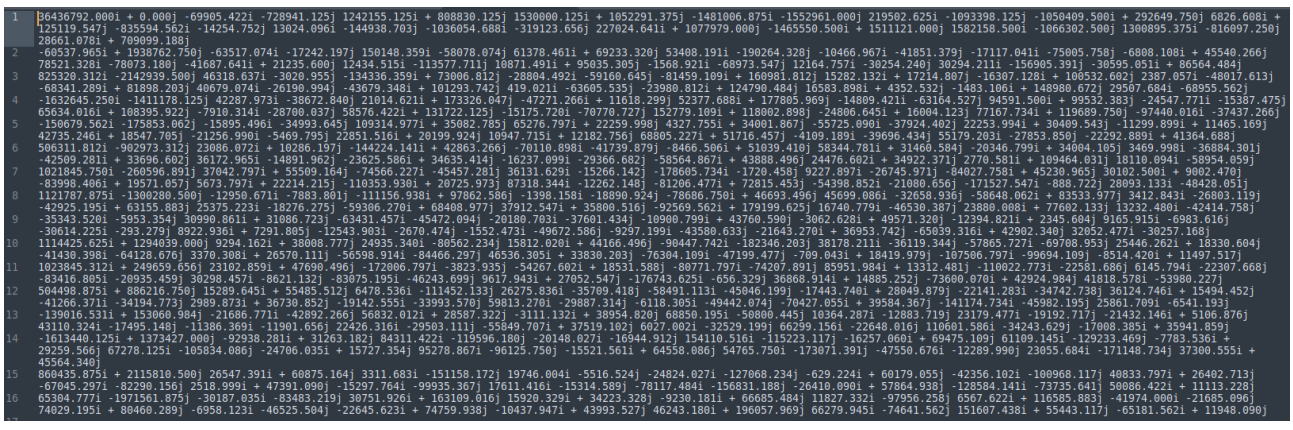
INPUT FILES AS:



SCREEN SHOTS FROM THE PROGRAMS:

-The time taken to find the 2d dft of the given matrix is proportional to the size of the matrix as we can observe in below test runs, as the matirx get's bigger the time complexity increases.





```
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$ vg ./hw5 -i inputFile1.txt -j inputFile2.txt -o outputFile.csv -n 4 -m 4
[Wed May 18 18:42:06] Two matrices of size 16x16 have read. The number of threads is 4
[Wed May 18 18:42:06] Thread 1 has reached the rendezvous point in 0.001443 seconds.
[Wed May 18 18:42:06] Thread 3 has reached the rendezvous point in 0.000267 seconds.
[Wed May 18 18:42:06] Thread 2 has reached the rendezvous point in 0.000162 seconds.
[Wed May 18 18:42:06] Thread 0 has reached the rendezvous point in 0.000166 seconds.
[Wed May 18 18:42:06] Thread 0 is advancing to the second part
[Wed May 18 18:42:06] Thread 2 is advancing to the second part
[Wed May 18 18:42:06] Thread 0 has finished the second part in 0.067398 seconds.
[Wed May 18 18:42:06] Thread 2 has finished the second part in 0.117793 seconds.
[Wed May 18 18:42:06] Thread 1 is advancing to the second part
[Wed May 18 18:42:06] Thread 1 has finished the second part in 0.051903 seconds.
[Wed May 18 18:42:06] Thread 3 is advancing to the second part
[Wed May 18 18:42:06] Thread 3 has finished the second part in 0.052259 seconds.
[Wed May 18 18:42:06] The process has written the output file. The Total time spent is 0.531694 seconds.
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$
```

```
36436792.000i + 0.000j -69905.422i -728941.125j 1242155.125i + 808830.125j 1530000.125i + 1052291.375j -1481006.875i -1552961.000j 219502.625i -1093398.125j -1050409.500i + 292649.750j 6826.608i +
125119.547j -835594.562i -14254.752j 13024.096i -144938.703j -1036054.688i -319123.656j 227024.641i + 1077979.000j -1465550.500i + 1511121.000j 1582158.500i -1066302.500j 1300895.375i -816097.250j
28661.078i + 709099.188j
-60537.965i + 1938762.750j -63517.074i -17242.197j 150148.359i -58078.074j 61378.461i + 69233.320j 53408.191i -190264.328j -10466.967i -41851.379j -17117.041i -75005.758j -6808.108i + 45540.266j
78521.328i -78073.180j -41687.641i + 21235.600j 12434.515i -113577.711j 10871.491i + 95035.305j -1568.921i -68973.547j 12164.757i -30254.240j 30294.211i -156905.391j -30595.051i + 86564.484j
825320.312i -2142939.500j 46318.637i -3020.955j -134336.359i + 73006.812j -28804.492i -59160.645j -81459.109i + 160981.812j 15282.132i + 17214.807j -16307.128i + 100532.602j 2387.057i -48017.613j
-68341.289i + 81898.281j 40679.074i -26190.994j -43679.348i + 101293.742j 419.021i -63605.535j -23980.812i + 124790.484j 16583.898i + 4352.532j -1483.106i + 148980.672j 29507.684i -68955.562j
-1632645.250i -1411178.125j 42287.973i -38672.840j 21014.621i + 173326.047j -47271.266i + 11618.299j 52377.688i + 177805.969j -14809.421i -63164.527j 94591.500i + 99532.383j -24547.771i -15387.475j
65634.016i + 108395.922j -7910.314i -28700.037j 58576.422i + 131722.125j -15175.720i -70770.727j 152779.109i + 118002.898j -24806.645i + 16004.123j 77167.734i + 119689.750j -97440.016i -37437.266j
-150679.562i -175853.062j -15895.496i -34993.645j 109314.977i + 35082.785j 65276.797i + 22259.998j 4327.755i + 34001.867j -55725.090i -37924.402j 22253.994i + 30409.543j -11299.899i + 11465.169j
42735.246i + 18547.705j -21256.990i -5469.795j 22851.516i + 20199.924j 10947.715i + 12182.756j 68805.227i + 51716.457j -4109.189i -39696.434j 55179.203i -27853.850j -22292.889i + 41364.688j
506311.812i -902973.312j 23086.072i + 10286.197j -144224.141i + 42863.266j -70110.898i -41739.879j -8466.506i + 51039.410j 58344.781i + 31460.584j -20346.799i + 34004.105j 3469.998i -36884.301j
-42589.281i + 33696.602j 36172.965i -14891.962j -23625.586i + 34635.414j -16237.099i -29366.682j -58564.867i + 43888.496j 24476.602i + 34922.371j 2770.581i + 109464.031j 18110.094i -58954.059j
1021845.750i -260596.891j 37042.797i + 55509.164j -74566.227i -45457.281j 36131.629i -15266.142j -178605.734i -1720.458j 9227.897i -26745.971j -84027.758i + 45230.965j 30102.500i + 9002.470j
-83998.406i + 19571.057j 5673.797i + 22214.215j -110353.930i + 20725.973j 87318.344i -12262.148j -81206.477i + 72815.453j -54398.852i -21080.656j -171527.547i -888.722j 28093.133i -48428.051j
1121787.875i -1300280.500j -12950.671i -7883.801j -111156.938i + 97862.586j -1398.158i -18890.924j -78686.750i + 46693.496j 45699.086i -32658.936j -58648.062i + 83533.977j 3412.843i -26803.119j
-42925.195i + 63155.883j 25375.223i -18276.275j -59306.270i + 68408.977j 37912.547i + 35800.516j -92569.562i + 179199.625j 16740.779i -46530.387j 23880.008i + 77602.133j 13232.480i -42414.758j
-35343.520i -5953.354j 30990.861i + 31086.723j -63431.457i -45472.094j -20180.703i -37601.434j -10900.799i + 43760.590j -3062.628i + 49571.320j -12394.821i + 2345.604j 9165.915i -6983.616j
-30614.225i -293.279j 8922.936i + 7291.805j -12543.903i -2670.474j -1552.473i -49672.586j -9297.199i -43580.633j -21643.270i + 36953.742j -65039.316i + 42902.340j 32052.477i -30257.168j
1114425.625i + 1294039.000j 9294.162i + 38008.777j 24935.340i -80562.234j 15812.020i + 44166.496j -90447.742i -182346.203j 38178.211i -36119.344j -57865.727i -69708.953j 25446.262i + 18330.604j
-41430.398i -64128.676j 3370.308i + 26570.111j -56598.914i -84466.297j 46536.305i + 33830.203j -76304.109i -47199.477j -709.043i + 18419.979j -107506.797i -99694.109j -8514.420i + 11497.517j
1023845.312i + 249659.656j 23102.859i + 47690.496j -172046.797i -3823.935j -54267.602i + 18531.588j -80771.797i -74207.891j 85951.984i + 13312.481j -110022.773i -22581.606j 6145.794i -22307.668j
-83416.805i -20935.459j 30298.457i -8621.132j -83075.195i -46243.699j 9617.943i + 27052.547j -176743.625i -656.329j 36868.914i + 14885.252j -73600.070i + 42924.984j 41818.578i -53980.227j
504498.875i + 886216.750j 15289.645i + 55485.512j 6478.536i -111452.133j 26275.836i -35709.418j -58491.113i -45046.199j -17443.740i + 28049.879j -22141.283i -34742.738j 36124.746i + 15494.452j
-41266.371i -34194.773j 2989.873i + 36730.852j -19142.555i -33993.570j 59813.270i -29887.314j -6118.305i -49442.074j -70427.055i + 39584.367j -141174.734i -45982.195j 25861.709i -6541.193j
-139016.531i + 153060.984j -21686.771i -42892.266j 56832.012i + 28587.322j -3111.132i + 38954.820j 68850.195i -50800.445j 10364.287i -12883.719j 23179.477i -19192.717j -21432.146i + 5106.876j
43110.324i -17495.148j -11386.369i -11901.656j 22426.316i -29503.111j -55849.707i + 37519.102j 6027.002i -32529.199j 66299.156i -22648.016j 110601.586i -34243.629j -17008.385i + 35941.859j
-1613440.125i + 1373427.000j -92938.281i + 31263.182j 84311.422i -119596.180j -20148.027i -16944.912j 154110.516i -115223.117j -16257.060i + 69475.109j 61109.145i -129233.469j -7783.5361 +
29259.566j 67278.125i -105834.086j -24706.035i + 15727.354j 95278.867i -96125.750j -15521.561i + 64558.086j 54765.750i -173071.391j -47550.676i -12289.990j 23055.684i -171148.734j 37300.555i +
45564.340j
860435.875i + 2115810.500j 26547.391i + 60875.164j 3311.683i -151158.172j 19746.004i -5516.524j -24824.027i -127068.234j -629.224i + 60179.055j -42356.102i -100968.117j 40833.797i + 26402.713j
-67045.297i -82290.156j 2518.999i + 47391.090j -15297.764i -99935.367j 17611.416i -15314.589j -78117.484i -156831.188j -26410.090i + 57864.938j -128584.141i -73735.641j 50086.422i + 11113.228j
65304.777i -1971561.875j -30187.035i -83483.219j 30751.926i + 163109.016j 15920.329i + 34223.328j -9230.181i + 66685.484j 11827.332i -97956.258j 6567.622i + 116585.883j -41974.000i -21685.096j
74029.195i + 80460.289j -6958.123i -46525.504j -22645.623i + 74759.938j -10437.947i + 43993.527j 46243.180i + 196057.969j 66279.945i -74641.562j 151607.438i + 55443.117j -65181.562i + 11948.090j
```

```
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$ vg ./hw5 -i inputFile1.txt -j inputFile2.txt -o outputFile.csv -n 5 -m 4
[Wed May 18 18:43:31] Two matrices of size 32x32 have read. The number of threads is 4
[Wed May 18 18:43:31] Thread 3 has reached the rendezvous point in 0.002528 seconds.
[Wed May 18 18:43:31] Thread 0 has reached the rendezvous point in 0.001376 seconds.
[Wed May 18 18:43:31] Thread 1 has reached the rendezvous point in 0.001325 seconds.
[Wed May 18 18:43:31] Thread 2 has reached the rendezvous point in 0.001260 seconds.
[Wed May 18 18:43:31] Thread 2 is advancing to the second part
[Wed May 18 18:43:31] Thread 2 has finished the second part in 0.853652 seconds.
[Wed May 18 18:43:31] Thread 1 is advancing to the second part
[Wed May 18 18:43:31] Thread 3 is advancing to the second part
[Wed May 18 18:43:31] Thread 3 has finished the second part in 0.843123 seconds.
[Wed May 18 18:43:31] Thread 1 has finished the second part in 1.687051 seconds.
[Wed May 18 18:43:31] Thread 0 is advancing to the second part
[Wed May 18 18:43:31] Thread 0 has finished the second part in 0.833082 seconds.
[Wed May 18 18:43:31] The process has written the output file. The Total time spent is 3.744896 seconds.
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$
```

```
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$ vg ./hw5 -i inputFile1.txt -j inputFile2.txt -o outputFile.csv -n 6 -m 4
[Wed May 18 18:50:17] Two matrices of size 64x64 have read. The number of threads is 4
[Wed May 18 18:50:18] Thread 1 has reached the rendezvous point in 0.011128 seconds.
[Wed May 18 18:50:18] Thread 2 has reached the rendezvous point in 0.009742 seconds.
[Wed May 18 18:50:18] Thread 3 has reached the rendezvous point in 0.009538 seconds.
[Wed May 18 18:50:18] Thread 0 has reached the rendezvous point in 0.009771 seconds.
[Wed May 18 18:50:18] Thread 0 is advancing to the second part
[Wed May 18 18:50:18] Thread 3 is advancing to the second part
[Wed May 18 18:50:18] Thread 2 is advancing to the second part
[Wed May 18 18:50:18] Thread 3 has finished the second part in 17.748315 seconds.
[Wed May 18 18:50:18] Thread 1 is advancing to the second part
[Wed May 18 18:50:18] Thread 0 has finished the second part in 40.605077 seconds.
[Wed May 18 18:50:18] Thread 1 has finished the second part in 24.643400 seconds.
[Wed May 18 18:50:18] Thread 2 has finished the second part in 52.513077 seconds.
[Wed May 18 18:50:17] The process has written the output file. The Total time spent is 55.690758 seconds.
ashraf@ashraf:~/Desktop/SEMESTER 10/2 SYSTEMS PROGRAMMING/HomeWorks/hw5/source_code$
```