Curtin University of Technology

School of Computing

Assignment Cover Sheet

(Please fill in all fields clearly and in upper-case letters)

Surname:	
Given Names:	
Student Number:	
Unix Username:	
Unit Name:	
Unit Number:	
Lecturer's Name:	
Tutor's Name:	
Assignment Title/Number:	
Due Date:	
Date Submitted:	

I declare the above information to be true, complete and correct.

Except where clearly indicated in this assignment, I hereby declare this assignment is solely my own work and has not been submitted for assessment or feedback purposes in this or any other unit whether at Curtin University of Technology or at any other University/Educational Institution.

I understand there are severe penalties for cheating, collusion and copying and I have read and understood the terms and conditions listed in the Unit Outline for this Unit.

Signed: _____ Date: _____

```c
/*****************************************************************************
** FILE: pmms.h
** AUTHOR: Brendan Lally
** STUDENT ID: 18407220
** UNIT: COMP2006 (Operating Systems)
** LAST MOD: 07/05/16
*****************************************************************************
**/

#ifndef PMMS_HEADER
#define PMMS_HEADER

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/mman.h>
#include <unistd.h>
#include <sys/types.h>
#include <semaphore.h>
#include <signal.h>

#define FALSE 0
#define TRUE !FALSE

/* Stores the semaphores and buffer for all the processes. */
typedef struct
{
    int subtotal; /*Subtotal of ith row*/
    pid_t process; /*Process ID*/
    sem_t mutex; /*Mutex semaphore*/
    sem_t empty; /*Producer semaphore*/
    sem_t full; /*Consumer semaphore*/
} Shared;

/*Function definitions*/
void* createMemory(char* name, int size);
void readMatrix(char* filename, int rows, int columns, int* matrix);
void printMatrix(int* matrix, int rows, int columns);
int getIndex(int rows, int columns, int ncols);
void childProcess(int* matrixA, int* matrixB, int* matrixC, Shared* ptr, int
process, int m, int n, int k);

#endif
```

```c
/*****************************************************************************
** FILE: pmms.c
** AUTHOR: Brendan Lally
** STUDENT ID: 18407220
** UNIT: COMP2006 (Operating Systems)
** PURPOSE: Imports two matrices and calculates the product using parallel
**          processes while summing each value into a row subtotal and
**          overall total. POSIX sempahores are used for process synchronization.
** LAST MOD: 07/05/16
*****************************************************************************
**/

#include "pmms.h"

int main(int argc, char *argv[])
{
    int i, j, m, n, k, pid, total = 0, errorCheck = FALSE;
    /* 2D integer arrays to point to shared memory block. */
    int *matrixA_ptr = NULL, *matrixB_ptr = NULL, *matrixC_ptr = NULL;
    /* Names of the shared memory objects. */
    char *matrixA = "matrixA", *matrixB = "matrixB", *matrixC = "matrixC",
        *subtotal = "subtotal";
    Shared *ptr;

    /* Error check the number of command line arguements. */
    if (argc != 6)
    {
        printf("Incorrect number of command line arguements!\n");
        errorCheck = TRUE;
    }

    /* If there was the correct number of command line arguments
       then continue with the program. */
    if (errorCheck != TRUE)
    {
        /* Convert command-line args from string to int. */
        m = atoi(argv[3]);
        n = atoi(argv[4]);
        k = atoi(argv[5]);

        if (m < 0 || n < 0 || k < 0)
        {
            printf("One or more matrix dimensions are invalid!\n");
            return 0;
        }

        /* Create shared memory objects. */
        matrixA_ptr = (int*)createMemory(matrixA, sizeof(int)*m*n);
        matrixB_ptr = (int*)createMemory(matrixB, sizeof(int)*n*k);
        matrixC_ptr = (int*)createMemory(matrixC, sizeof(int)*m*k);
        ptr = (Shared*)createMemory(subtotal, sizeof(Shared));

        /* Read matrix files. */
```

```
54          readMatrix(argv[1], m, n, matrixA_ptr);
55          readMatrix(argv[2], n, k, matrixB_ptr);
56
57          /* Initialise semaphores. */
58          sem_init(&ptr->mutex, 1, 1);
59          sem_init(&ptr->empty, 1, 1);
60          sem_init(&ptr->full, 1, 0);
61
62          /* Kills zombie processes. */
63          signal(SIGCHLD, SIG_IGN);
64
65          /* Create m child processes. */
66          for (i = 0; i < m; i++)
67          {
68              pid = fork();
69              /*Fork error*/
70              if (pid < 0)
71              {
72                  perror("Fork Failed");
73                  return 0;
74              }
75              /*Do child processing*/
76              else if (pid == 0)
77              {
78                  childProcess(matrixA_ptr, matrixB_ptr, matrixC_ptr, ptr, i, m, n,
                    k);
79                  exit(0);
80              }
81          }
82          /*Parent Process*/
83          for (j = 0; j < m; j++)
84          {
85              sem_wait(&ptr->full);
86              sem_wait(&ptr->mutex);
87              /*Consume buffer*/
88              printf("Subtotal produced by process with ID %d: %d\n",ptr->process,
                ptr->subtotal);
89              total += ptr->subtotal;
90              sem_post(&ptr->mutex);
91              sem_post(&ptr->empty);
92          }
93          printf("Total: %d\n", total);
94      }
95
96      return 0;
97  }
98
99  /* Creates a shared memory block and returns a pointer to that block. Imports
100 ** the name and size of a block to create.
101 */
102 void* createMemory(char* name, int size)
103 {
104     int shm_fd;
```

```
105       void* shm_ptr;
106
107       /* Create shared memory object. */
108       shm_fd = shm_open(name, O_RDWR | O_CREAT, 0666);
109
110       /* Configure the size of the shared memory object. */
111       ftruncate(shm_fd, size);
112
113       /* Memory map the shared object. */
114       shm_ptr = mmap(0, size, PROT_READ | PROT_WRITE, MAP_SHARED, shm_fd, 0);
115
116       close(shm_fd);
117
118       if (shm_ptr == MAP_FAILED)
119       {
120           printf("MEMORY MAP FAILED\n");
121           return 0;
122       }
123
124       return shm_ptr;
125   }
126
127   /* Reads the matrix from a file into shared memory. Imports the filename,
128   ** matrix dimensions and a pointer to the shared memory where the matrix
129   ** will be stored.
130   */
131   void readMatrix(char* filename, int rows, int columns, int* matrix)
132   {
133       int ii, jj;
134       FILE* fMatrix = fopen(filename, "r");
135
136       if (fMatrix == NULL)
137       {
138           perror("Error opening file");
139       }
140       else
141       {
142           for (ii = 0; ii < rows; ii++)
143           {
144               for(jj = 0; jj < columns; jj++)
145               {
146                   fscanf(fMatrix, "%d", &matrix[getIndex(ii, jj, columns)]);
147               }
148           }
149       }
150
151       fclose(fMatrix);
152   }
153
154   /* The producer function that each child process runs. Each child process
155   ** calculates the subtotal for its given row in the matrix. Imports pointers
156   ** to all the shared memory blocks along with the matrix dimensions and the
157   ** process number.
```

```
158  */
159  void childProcess(int* matrixA, int* matrixB, int* matrixC, Shared* ptr,
160      int processNum, int m, int n, int k)
161  {
162      int i, subtotal = 0;
163
164      for (i = 0; i < k; i++)
165      {
166
167          matrixC[getIndex(processNum, i, k)] =
168              matrixA[getIndex(processNum, 0, n)]*matrixB[getIndex(0, i, k)]
169                  + matrixA[getIndex(processNum, 1, n)]*matrixB[getIndex(1, i, k)];
170
171          subtotal += matrixC[getIndex(processNum, i, k)];
172      }
173
174      sem_wait(&ptr->empty);
175      sem_wait(&ptr->mutex);
176      /*Produce subtotal to be consumed. Stores the process number and the subtotal*/
177      ptr->process = getpid();
178      ptr->subtotal = subtotal;
179      sem_post(&ptr->mutex);
180      sem_post(&ptr->full);
181  }
182
183  /* Function to calculate the index of the 1D array that corresponds to a given
184  ** 2D array. Imports the rows, columns and number of columns for a 2D array and
185  ** returns the index.
186  */
187  int getIndex(int rows, int columns, int ncols)
188  {
189      return rows*ncols + columns;
190  }
```

```c
/*****************************************************************************
** FILE: pmmsThread.h
** AUTHOR: Brendan Lally
** STUDENT ID: 18407220
** UNIT: COMP2006 (Operating Systems)
** LAST MOD: 07/05/16
*****************************************************************************
**/

#ifndef PMMS_HEADER
#define PMMS_HEADER

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <fcntl.h>
#include <unistd.h>
#include <pthread.h>

#define FALSE 0
#define TRUE !FALSE

/* Stores the data required for each thread to compute the subtotal*/
typedef struct
{
    int subtotal; /*Subtotal of ith row*/
    pthread_t thread; /*Thread ID*/
    int* matrixA;
    int* matrixB;
    int* matrixC;
    int m; /*Number of rows for matrix A and matrix C*/
    int n; /*Number of columns for matrix A and rows for matrix B*/
    int k; /*Number of columns for matrix B and matrix C*/
} Shared;

/*Shared objects - Global variables*/
Shared* s;
pthread_mutex_t mutex;
pthread_cond_t empty; /*Producer variable*/
pthread_cond_t full; /*Consumer variable*/

/*Function definitions*/
void* createMemory(char* name, int size);
void readMatrix(char* filename, int rows, int columns, int* matrix);
void printMatrix(int* matrix, int rows, int columns);
int getIndex(int rows, int columns, int ncols);
void* calculateSubtotal(void* ptr);

#endif
```

```c
/*****************************************************************************
** FILE: pmmsThread.c
** AUTHOR: Brendan Lally
** STUDENT ID: 18407220
** UNIT: COMP2006 (Operating Systems)
** PURPOSE: Imports two matrices and calculates the product using multi-threading
**          while summing each value into a row subtotal and overall total.
**          POSIX threads are used for thread creation and synchronization.
** LAST MOD: 07/05/16
*****************************************************************************
**/

#include "pmmsThread.h"

int main(int argc, char *argv[])
{
    int i, j, total = 0, errorCheck = FALSE;
    pthread_t *tid; /*Thread Identifier */

    /* Error check the number of command line arguements. */
    if (argc != 6)
    {
        printf("Incorrect number of command line arguements!\n");
        errorCheck = TRUE;
    }
    /* If there was the correct number of command line arguments
       then continue with the program. */
    if (errorCheck != TRUE)
    {
        s = (Shared*)malloc(sizeof(Shared));

        /* Convert command-line args from string to int*/
        s->m = atoi(argv[3]);
        s->n = atoi(argv[4]);
        s->k = atoi(argv[5]);

        if (s->m < 0 || s->n < 0 || s->k < 0)
        {
            printf("One or more matrix dimensions are invalid!\n");
            return 0;
        }

        s->subtotal = 0;
        s->thread = 0;
        s->matrixA = (int*)malloc(sizeof(int)*s->m*s->n);
        s->matrixB = (int*)malloc(sizeof(int)*s->n*s->k);
        s->matrixC = (int*)malloc(sizeof(int)*s->m*s->k);

        readMatrix(argv[1], s->m, s->n, s->matrixA);
        readMatrix(argv[2], s->n, s->k, s->matrixB);

        tid = (pthread_t*)malloc(sizeof(pthread_t)*s->m);

```

```c
        /* Initialise mutex and condition variables */
        pthread_mutex_init(&mutex, NULL);
        pthread_cond_init(&empty, NULL);
        pthread_cond_init(&full, NULL);

        /* Create m threads*/
        for (i = 0; i < s->m; i++)
        {
            /*Thread creation error*/
            if (pthread_create(&tid[i], NULL, calculateSubtotal,
                (void *)(intptr_t)(i+1)) != 0)
            {
                printf("Thread Creation Failed!");
                exit(0);
            }
            /* Mark each thread so its resources are auto released when the
               thread terminates. */
            pthread_detach(tid[i]);
        }

        /*Consumer Process*/
        for (j = 1; j <= s->m; j++)
        {
            pthread_mutex_lock(&mutex);
            while (s->subtotal == 0)
            {
                pthread_cond_wait(&full, &mutex);
            }
            /*Consume buffer*/
            printf("Subtotal produced by thread with ID %lu: %d\n",
                (unsigned long)s->thread, s->subtotal);
            total += s->subtotal;
            s->subtotal = 0;
            pthread_cond_signal(&empty);
            pthread_mutex_unlock(&mutex);
        }
        printf("Total: %d\n", total);

        /* Clean up allocated memory. */
        free(tid);
        free(s->matrixA);
        free(s->matrixB);
        free(s->matrixC);
        free(s);

        pthread_mutex_destroy(&mutex);  /* Free up mutex */
        pthread_cond_destroy(&empty);   /* Free up producer condition variable */
        pthread_cond_destroy(&full);    /* Free up consumer condition variable */

    }
    return 0;
}

```

```
/* Reads the matrix from a file into memory. Imports the filename,
** matrix dimensions and a pointer to the shared memory where the matrix
** will be stored.
*/
void readMatrix(char* filename, int rows, int columns, int* matrix)
{
    int ii, jj;
    FILE* fMatrix = fopen(filename, "r");

    if (fMatrix == NULL)
    {
        perror("Error opening file");
    }
    else
    {
        for (ii = 0; ii < rows; ii++)
        {
            for(jj = 0; jj < columns; jj++)
            {
                fscanf(fMatrix, "%d", &matrix[getIndex(ii, jj, columns)]);
            }
        }
    }

    fclose(fMatrix);
}

/* The producer function that each created thread runs. Each producer thread
** calculates the subtotal for its given row in the matrix. Imports a pointer
** to the row number the thread is to calculate.
*/
void* calculateSubtotal(void* ptr)
{
    int i, subtotal = 0, process, n, k;

    /* Each thread should get its values mutually exclusively. */
    pthread_mutex_lock(&mutex);
    process = (intptr_t)ptr - 1;
    n = s->n;
    k = s->k;
    pthread_mutex_unlock(&mutex);


    for (i = 0; i < k; i++)
    {
        s->matrixC[getIndex(process, i, k)] =
            s->matrixA[getIndex(process, 0, n)]*s->matrixB[getIndex(0, i, k)]
                + s->matrixA[getIndex(process, 1, n)]*s->matrixB[getIndex(1, i, k)];

        subtotal += s->matrixC[getIndex(process, i, k)];
    }
    pthread_mutex_lock(&mutex);

```

```
160       while (s->subtotal != 0)
161       {
162           pthread_cond_wait(&empty, &mutex);
163       }
164       /* Produce subtotal to be consumed. Stores the thread id and the subtotal. */
165       s->thread = pthread_self();
166       s->subtotal = subtotal;
167
168       pthread_cond_signal(&full);
169       pthread_mutex_unlock(&mutex);
170
171       return 0;
172   }
173
174   /* Function to calculate the index of the 1D array that corresponds to a given
175   ** 2D array. Imports the rows, columns and number of columns for a 2D array and
176   ** returns the index.
177   */
178   int getIndex(int rows, int columns, int ncols)
179   {
180       return rows*ncols + columns;
181   }
182
```

Discussion

Processes

Setting up Shared memory

With the multi-process implementation, each matrix along with the data structure for the subtotal and semaphores is stored in a separate shared memory block created using POSIX shared memory. The shared memory objects were created using the *shm_open*() function, allowing read/write capabilities. Each shared memory object was then truncated to exactly the right size for each matrix based on the dimensions given in the command line parameters. The shared memory objects were then mapped to memory with the *mmap*() function using MAP_SHARED to indicate that changes to the shared memory object will be visible to all processes sharing the object.

Accessing Shared Memory and the Producer – Consumer problem

Each child process accesses the shared memory objects for matrix A and matrix B in order to calculate the values for matrix C. Each process also has access to the subtotal shared memory object. Mutual exclusion is handled by three semaphores: mutex, empty and full. The mutex semaphore is initialised to 1 and is used to ensure mutual exclusion while the child processes updates the subtotal and while the parent process prints the subtotal and calculates the overall total. It ensures mutual exclusion by only allowing 1 child process or parent process to update the shared data at any given time, removing the chance that race condition will occur. The other two semaphores; empty and full are used to correctly implement the producer consumer problem. The empty and full semaphores count the number of empty and full buffers and are initially set to 1 and 0 respectively.

The first child process will be able to update the subtotal data structure by calling *sem_wait*(empty) which will decrement the semaphore to 0. Any other child processes that want to update subtotal will also call *sem_wait*(empty) further decrementing empty, but instead they will be blocked due to the value of empty being less than 0.

Meanwhile, the parent process calls *sem_wait*(full) which will decrement full to less than 0 and block the parent process. Once a child process has successfully updated its subtotal, sem_post(full) is called which will increment full to 0, unblocking the parent process and allowing it to consume the recently updated subtotal. Once the parent has consumed the subtotal it calls *sem_post*(empty) which then allows the next child process that is waiting to update its subtotal.

This pattern of allowing only one child process to update subtotal before the parent process consumes it repeats until all the child processes have managed to update their respective subtotals and the parent process has consumed all of them and calculated the overall total.

Threads

Due to the nature of threads it was not necessary to implement the shared memory objects for the matrices and the subtotal data structure. It was necessary however to make them global variables to enable each thread to access them.

Accessing Shared Resources and the Producer – Consumer problem

The producer-consumer problem works the same as it does when using the multi-process implementation. M (number of rows in matrix C) threads are created which all calculate the subtotal for their respective rows. A mutex is used to provide mutual exclusion whilst producer threads and the consumer thread are accessing shared information. The empty and full semaphores from the multi-process implementation are represented as pthread condition variables which address communication between threads that share a mutex. Similarly to the semaphores, the empty and full condition variables alternate in the same pattern, allowing a producer thread to only add its subtotal if the subtotal buffer is empty, and the producer to only consume the subtotal if it's not empty, by blocking until the condition is met.

#README.txt

Steps to run Operating Systems Assignment.

****Using Processes****

1. Navigate to the target directory ~/OS/assignment
2. Run the makefile using the command 'make' to compile the program
3. Run the program using ./pmms matrixA matrixB M N K **

** where matrixA and matrixB are two text files that contain an M x N and an N x K matrix respectively.
M is the number of rows in matrixA
N is the number of columns in matrixA and rows in matrixB
K is the number of columns in matrixB

eg. ./pmms test2MatrixA test2MatrixB 20 11 40

****Using Threads****

1. Navigate to the target directory ~/OS/assignment/Threads
2. Run the makefile using the command 'make' to compile the program
3. Run the program using ./pmmsThread matrixA matrixB M N K **

** where matrixA and matrixB are two text files that contain an M x N and an N x K matrix respectively.
M is the number of rows in matrixA
N is the number of columns in matrixA and rows in matrixB
K is the number of columns in matrixB

eg. ./pmms test4MatrixA test4MatrixB 100 100 100

test1MatrixA

1 2
3 4
5 6

test1MatrixB

1 2 3 4
5 6 7 8

test2MatrixA

11 20 12 9 16 5 18 17 13 6 14
5 12 2 1 9 4 10 13 16 11 19
6 14 9 11 3 12 17 20 7 4 16
5 7 11 3 2 13 8 12 6 16 19
7 16 2 3 17 1 8 19 6 9 14
7 10 19 2 9 5 14 12 18 3 20
4 20 7 14 12 6 13 18 3 5 8
11 15 10 6 9 14 4 7 12 5 17
11 14 12 19 15 7 17 13 4 16 2
4 3 14 19 18 8 1 6 5 11 10
7 2 10 18 16 19 3 8 4 20 12
12 13 18 3 19 6 1 16 8 17 2
17 4 9 5 2 3 14 18 10 16 8
10 18 13 3 1 5 14 16 6 2 11
10 15 9 17 7 12 5 4 6 20 16
7 11 15 12 13 10 9 3 4 19 1
14 9 11 8 20 12 1 10 7 18 3
5 20 11 18 2 3 7 12 16 8 13
5 12 1 13 20 11 16 10 15 14 9
20 3 16 9 2 4 11 5 13 18 19

test2MatrixB

12 9 34 11 31 28 25 26 24 37 3 14 21 19 10 17 6 38 13 7 8 29 15 16 39 4 20 36 5 18 33 30
32 1 23 22 35 2 27 40
4 37 14 5 38 22 15 40 24 28 35 12 23 36 32 2 7 26 17 10 39 25 18 21 6 13 30 31 34 29 20 1
11 33 9 3 19 27 16 8
28 23 10 40 36 16 1 38 7 6 31 2 39 34 14 24 9 8 17 20 4 21 11 37 12 5 35 15 26 27 30 22 3
33 18 25 29 32 13 19
32 5 19 7 18 3 2 16 26 29 40 4 37 10 14 8 33 24 11 12 36 38 22 34 21 25 20 39 17 1 27 30 6
15 35 23 31 28 13 9
4 5 25 23 30 10 15 17 28 40 18 14 34 7 11 29 6 31 8 22 38 36 13 9 35 3 37 24 26 20 39 12
19 21 27 16 1 32 2 33
33 38 24 28 18 34 10 13 4 6 26 1 20 31 3 30 2 8 15 5 23 29 25 40 11 27 36 32 14 35 37 21
39 16 19 12 22 9 7 17
36 26 30 17 14 4 12 2 35 32 28 18 25 15 37 11 22 27 31 5 3 34 9 38 40 21 16 33 8 23 24 19
10 39 13 7 1 29 6 20

10 39 18 37 4 26 7 11 31 9 23 16 28 32 38 35 25 22 34 8 2 13 20 21 40 19 17 36 15 24 3 27
6 33 29 30 5 14 1 12
10 38 36 25 6 27 21 37 13 9 2 19 17 15 12 1 39 5 16 32 8 7 33 40 4 3 28 34 20 26 11 18 14
22 31 23 24 29 35 30
7 5 34 4 18 3 11 20 8 14 17 33 16 15 2 1 35 25 19 40 37 26 31 23 12 32 27 22 38 29 21 9 13
24 6 36 30 10 39 28
18 13 37 6 34 2 8 40 25 31 23 39 19 7 1 9 12 5 3 11 14 32 10 30 20 33 16 21 28 22 27 26 24
35 36 29 38 17 15 4

test3MatrixA

9 14 11 13 7 20 6 8 17 5 4 19 18 10 3 21 1 2 15 12 16
2 14 10 5 9 12 13 16 17 7 19 1 15 11 4 18 21 20 8 3 6
6 1 5 14 21 8 4 2 12 20 7 9 15 3 18 10 17 13 11 16 19
16 10 3 15 5 17 13 6 20 9 2 7 11 4 12 21 14 1 19 18 8
10 8 7 1 15 6 4 3 17 21 2 14 16 11 19 18 13 9 12 20 5
1 16 11 20 15 14 18 5 3 10 7 6 13 21 2 4 8 17 19 12 9
9 17 10 7 14 21 2 19 13 20 1 3 6 12 16 15 4 11 8 5 18
18 1 17 10 19 11 16 7 14 20 13 15 21 2 5 3 4 6 12 9 8
19 11 16 6 5 17 12 21 13 9 15 18 8 3 2 14 4 20 10 1 7
10 11 13 6 8 4 15 17 7 1 9 3 21 19 20 18 12 2 5 16 14
15 2 19 9 14 11 3 16 1 8 5 20 4 18 10 17 12 6 7 13 21
19 13 10 3 18 14 1 17 2 20 12 9 5 16 4 11 6 8 7 15 21
5 18 13 4 20 21 9 17 19 10 12 15 6 2 11 14 7 3 8 16 1
1 15 13 21 16 5 3 9 17 20 14 2 19 8 4 18 7 10 11 6 12
18 21 5 8 19 1 13 12 7 16 4 20 11 14 9 10 17 15 6 2 3
16 20 15 17 21 6 9 7 11 8 4 13 19 5 2 18 12 14 1 10 3
17 3 4 20 16 2 15 13 18 8 6 14 7 11 12 9 1 10 19 5 21
12 17 15 11 10 6 4 14 7 2 3 18 19 21 9 8 13 1 5 16 20
21 18 12 3 10 15 2 19 17 6 14 1 4 5 13 9 11 8 16 20 7
10 4 8 2 21 15 16 7 5 9 1 18 3 14 12 13 20 19 11 6 17
10 3 12 4 11 16 18 1 7 6 15 20 13 9 14 21 17 2 8 19 5
18 20 10 14 17 1 6 7 13 21 15 8 11 2 19 3 16 5 12 4 9
21 17 12 4 10 6 9 13 19 7 3 11 18 14 15 16 2 8 5 20 1
2 20 8 3 4 10 11 12 19 6 9 7 1 18 14 13 21 15 5 16 17
7 1 19 18 5 13 11 17 16 12 6 20 15 9 8 2 21 10 4 3 14
9 21 16 17 10 13 12 3 11 7 19 20 5 14 1 6 18 15 2 4 8
11 9 6 12 2 7 8 15 13 18 17 1 20 3 10 21 14 4 16 19 5
20 15 18 9 14 16 19 8 2 5 4 11 10 17 21 7 6 13 12 3 1
2 5 3 4 12 15 21 17 19 8 7 6 13 18 10 1 9 14 20 11 16
13 5 12 10 1 18 6 17 8 11 21 14 9 4 7 20 19 2 16 15 3
11 15 14 2 1 16 3 20 10 4 18 13 8 19 12 17 9 21 6 5 7
1 14 7 4 5 15 8 21 13 12 2 18 11 17 3 10 9 20 19 16 6
12 13 4 7 2 17 9 8 14 11 19 21 18 3 6 5 20 10 1 15 16
3 20 1 6 7 4 8 21 2 14 16 18 13 10 15 12 11 19 17 9 5
8 16 17 13 5 19 9 4 6 11 7 3 10 14 15 18 21 12 20 2 1
13 10 17 19 11 14 6 12 21 4 8 20 5 9 2 3 1 7 16 18 15
2 21 12 4 19 18 16 20 6 1 15 11 17 9 8 7 13 3 5 14 10
19 2 16 20 18 4 3 14 13 12 1 5 7 6 11 10 9 8 15 21 17
15 9 10 6 3 7 11 1 20 8 14 12 18 4 17 2 19 13 5 21 16
9 2 4 20 10 12 17 16 18 7 6 13 15 3 1 19 21 5 14 8 11
15 10 14 3 12 2 6 17 19 9 11 4 1 13 7 18 21 16 5 8 20

14 10 5 11 9 15 18 13 4 16 21 20 8 12 2 6 3 17 19 1 7
7 21 6 2 11 16 1 18 19 15 12 20 4 10 17 9 14 3 8 5 13
5 3 7 11 10 2 15 4 16 6 8 18 19 20 17 1 9 12 14 13 21
21 11 10 17 2 14 4 19 12 13 5 6 3 15 7 18 1 9 16 8 20
11 16 12 21 2 19 4 15 20 1 13 18 10 6 7 14 3 5 8 17 9
11 2 21 17 12 6 16 3 14 9 5 10 19 13 15 1 7 8 20 18 4
3 11 2 20 9 16 17 5 8 14 10 15 4 21 13 1 19 7 6 12 18
10 18 16 3 17 8 20 5 19 14 1 12 11 9 13 15 6 2 21 4 7
1 2 6 7 18 5 4 15 9 17 13 10 14 21 19 20 12 3 11 16 8
12 17 9 4 11 21 15 3 20 5 2 7 1 19 18 8 10 13 14 6 16
19 13 12 21 3 4 20 6 8 16 18 11 17 10 15 7 5 2 9 1 14
16 11 10 18 20 2 14 19 15 1 17 9 5 3 4 8 21 7 13 6 12
12 11 2 20 5 10 3 18 13 16 6 14 19 8 7 17 1 4 15 9 21
6 20 5 14 2 4 13 11 9 12 21 18 10 15 17 8 16 7 3 1 19
1 4 20 19 15 16 12 5 6 8 18 9 3 2 14 11 10 17 13 21 7
3 16 11 13 5 1 20 21 14 18 7 12 15 2 8 6 4 17 9 10 19
18 15 14 17 19 12 3 4 1 11 10 20 7 2 9 5 6 21 8 13 16
16 7 12 2 21 14 20 11 8 5 10 17 1 19 13 3 6 9 4 18 15
5 2 12 1 10 13 20 18 11 9 16 21 6 7 19 17 14 3 4 15 8
7 4 5 13 1 2 19 11 8 3 20 16 15 17 14 6 12 21 10 18 9
8 21 4 11 19 16 2 14 9 7 1 10 13 12 17 6 18 3 20 5 15
14 21 20 8 9 19 13 3 5 15 12 18 4 17 7 10 16 11 6 2 1
16 17 11 18 6 9 20 12 1 13 15 8 3 14 2 4 19 7 10 5 21
14 16 5 20 10 2 3 18 11 21 8 12 13 6 9 1 7 19 15 17 4
1 20 11 12 2 9 14 5 21 10 13 19 7 16 18 17 6 3 15 4 8
11 13 20 2 17 8 14 6 1 7 15 3 9 16 19 10 5 18 21 4 12
11 20 12 17 9 2 4 1 8 5 19 13 18 21 6 14 3 15 16 10 7
1 13 14 2 4 18 19 11 20 17 7 5 6 10 15 16 12 8 9 3 21
18 9 6 21 20 16 3 7 10 17 5 8 14 19 15 4 11 12 2 13 1
8 20 6 10 21 9 14 17 12 16 4 5 1 11 2 13 7 15 18 3 19
10 11 17 3 1 9 15 21 4 13 12 5 14 8 7 19 16 20 18 2 6
11 2 4 10 5 12 9 8 19 21 18 6 13 20 17 14 15 16 7 3 1
17 1 12 19 7 10 16 2 3 11 21 9 4 15 6 13 14 20 18 8 5
18 4 7 15 12 5 1 2 16 13 8 6 19 20 11 21 17 9 14 3 10
2 10 3 5 19 11 12 7 9 20 4 14 21 16 13 1 17 18 6 15 8
16 19 11 6 3 17 18 9 15 21 4 2 13 20 1 10 5 14 8 12 7
9 14 19 13 2 1 12 8 18 16 5 10 17 21 6 7 11 4 20 15 3
11 15 13 12 1 3 21 19 17 18 7 8 9 10 6 16 4 20 5 2 14
2 1 15 17 19 20 7 21 14 11 10 12 3 13 6 9 18 5 16 8 4
6 19 16 14 12 2 13 3 11 5 17 1 4 7 15 8 9 18 21 10 20
8 7 18 20 6 19 5 12 2 13 14 3 17 21 9 16 10 15 1 4 11
5 12 21 16 1 13 3 4 6 17 10 9 20 2 19 15 11 7 8 14 18
9 17 6 21 13 19 8 3 18 16 7 20 2 1 4 5 15 10 14 11 12
19 17 2 21 8 10 14 5 6 16 1 9 12 18 15 13 11 20 4 3 7
12 3 2 9 20 8 17 10 1 5 19 7 11 4 18 14 21 16 15 6 13
9 20 12 3 15 6 4 11 8 2 13 19 18 21 7 14 10 1 5 16 17
12 7 19 11 6 18 14 10 15 16 13 1 20 21 9 5 17 3 2 8 4
3 2 11 1 7 16 9 18 4 6 5 13 10 14 21 17 12 19 15 8 20
13 17 8 21 19 15 9 2 10 5 4 16 7 20 18 3 6 12 14 1 11
12 21 15 9 16 17 6 3 8 20 1 18 7 13 10 19 11 14 5 4 2
13 16 7 14 12 17 8 21 2 20 3 19 6 9 1 11 5 10 15 4 18
5 2 6 17 10 15 12 21 8 4 19 14 16 1 9 18 7 11 3 13 20

test3MatrixB

11 16 12 1 10 14
12 15 4 21 6 3
10 4 21 7 12 1
14 19 10 17 2 7
8 13 1 9 18 21
20 8 10 12 15 21
18 21 20 19 13 7
19 1 18 2 5 7
12 18 7 6 16 19
16 9 13 7 1 21
13 8 7 14 16 15
7 2 12 3 10 4
8 14 2 19 17 20
21 3 5 8 9 1
15 9 16 12 20 6
9 11 1 17 4 20
16 5 2 4 8 12
20 5 8 18 11 16
11 3 2 8 14 6
6 15 21 1 2 4
21 9 3 8 12 4

test4MatrixA and test4MatrixB are just 100x100 matrices with values of 1

```
TEST 1 RESULTS
./pmms test1MatrixA test1MatrixB 3 2 4

Subtotal produced by process with ID 13018: 62
Subtotal produced by process with ID 13019: 134
Subtotal produced by process with ID 13020: 206
Total: 402



TEST 2 RESULTS
./pmms test2MatrixA test2MatrixB 20 11 40

Subtotal produced by process with ID 13026: 25420
Subtotal produced by process with ID 13027: 13940
Subtotal produced by process with ID 13028: 16400
Subtotal produced by process with ID 13029: 9840
Subtotal produced by process with ID 13030: 18860
Subtotal produced by process with ID 13031: 13940
Subtotal produced by process with ID 13032: 19680
Subtotal produced by process with ID 13033: 21320
Subtotal produced by process with ID 13034: 20500
Subtotal produced by process with ID 13035: 5740
Subtotal produced by process with ID 13036: 7380
Subtotal produced by process with ID 13037: 20500
Subtotal produced by process with ID 13038: 17220
Subtotal produced by process with ID 13039: 22960
Subtotal produced by process with ID 13040: 20500
Subtotal produced by process with ID 13041: 14760
Subtotal produced by process with ID 13042: 18860
Subtotal produced by process with ID 13043: 20500
Subtotal produced by process with ID 13044: 13940
Subtotal produced by process with ID 13045: 18860
Total: 341120



TEST 3 RESULTS
./pmms test3MatrixA test3MatrixB 93 21 6

Subtotal produced by process with ID 13051: 1430
Subtotal produced by process with ID 13052: 982
Subtotal produced by process with ID 13053: 445
Subtotal produced by process with ID 13054: 1634
Subtotal produced by process with ID 13055: 1128
Subtotal produced by process with ID 13056: 1040
Subtotal produced by process with ID 13057: 1613
Subtotal produced by process with ID 13058: 1213
Subtotal produced by process with ID 13059: 1887
Subtotal produced by process with ID 13060: 1311
Subtotal produced by process with ID 13061: 1082
Subtotal produced by process with ID 13062: 2009
Subtotal produced by process with ID 13063: 1418
```

Subtotal produced by process with ID 13064: 979
Subtotal produced by process with ID 13065: 2433
Subtotal produced by process with ID 13066: 2244
Subtotal produced by process with ID 13067: 1271
Subtotal produced by process with ID 13068: 1805
Subtotal produced by process with ID 13069: 2442
Subtotal produced by process with ID 13070: 884
Subtotal produced by process with ID 13071: 823
Subtotal produced by process with ID 13072: 2372
Subtotal produced by process with ID 13073: 2381
Subtotal produced by process with ID 13074: 1348
Subtotal produced by process with ID 13075: 509
Subtotal produced by process with ID 13076: 1857
Subtotal produced by process with ID 13077: 1253
Subtotal produced by process with ID 13078: 2195
Subtotal produced by process with ID 13079: 433
Subtotal produced by process with ID 13080: 1137
Subtotal produced by process with ID 13081: 1619
Subtotal produced by process with ID 13082: 918
Subtotal produced by process with ID 13083: 1561
Subtotal produced by process with ID 13084: 1412
Subtotal produced by process with ID 13085: 1488
Subtotal produced by process with ID 13086: 1442
Subtotal produced by process with ID 13087: 1409
Subtotal produced by process with ID 13088: 1338
Subtotal produced by process with ID 13089: 1509
Subtotal produced by process with ID 13090: 698
Subtotal produced by process with ID 13091: 1570
Subtotal produced by process with ID 13092: 1506
Subtotal produced by process with ID 13093: 1729
Subtotal produced by process with ID 13094: 503
Subtotal produced by process with ID 13095: 2015
Subtotal produced by process with ID 13096: 1680
Subtotal produced by process with ID 13097: 826
Subtotal produced by process with ID 13098: 863
Subtotal produced by process with ID 13099: 1738
Subtotal produced by process with ID 13100: 186
Subtotal produced by process with ID 13101: 1805
Subtotal produced by process with ID 13102: 2009
Subtotal produced by process with ID 13103: 1695
Subtotal produced by process with ID 13104: 1439
Subtotal produced by process with ID 13105: 1604
Subtotal produced by process with ID 13106: 308
Subtotal produced by process with ID 13107: 1168
Subtotal produced by process with ID 13108: 2067
Subtotal produced by process with ID 13109: 1451
Subtotal produced by process with ID 13110: 442
Subtotal produced by process with ID 13111: 692
Subtotal produced by process with ID 13112: 1793
Subtotal produced by process with ID 13113: 2177
Subtotal produced by process with ID 13114: 2061
Subtotal produced by process with ID 13115: 1872
Subtotal produced by process with ID 13116: 1284

```
Subtotal produced by process with ID 13117: 1497
Subtotal produced by process with ID 13118: 1924
Subtotal produced by process with ID 13119: 857
Subtotal produced by process with ID 13120: 1701
Subtotal produced by process with ID 13122: 1311
Subtotal produced by process with ID 13123: 826
Subtotal produced by process with ID 13125: 1396
Subtotal produced by process with ID 13126: 738
Subtotal produced by process with ID 13127: 2183
Subtotal produced by process with ID 13128: 1430
Subtotal produced by process with ID 13129: 1619
Subtotal produced by process with ID 13121: 1732
Subtotal produced by process with ID 13130: 189
Subtotal produced by process with ID 13131: 1543
Subtotal produced by process with ID 13132: 939
Subtotal produced by process with ID 13133: 1052
Subtotal produced by process with ID 13124: 1149
Subtotal produced by process with ID 13134: 1613
Subtotal produced by process with ID 13135: 2253
Subtotal produced by process with ID 13136: 951
Subtotal produced by process with ID 13137: 1796
Subtotal produced by process with ID 13138: 1195
Subtotal produced by process with ID 13139: 314
Subtotal produced by process with ID 13140: 1869
Subtotal produced by process with ID 13141: 2049
Subtotal produced by process with ID 13142: 1808
Subtotal produced by process with ID 13143: 442
Total: 129811



TEST 4 RESULTS
./pmms test4MatrixA test4MatrixB 100 100 100

Subtotal produced by process with ID 13149: 200
Subtotal produced by process with ID 13248: 200
Subtotal produced by process with ID 13151: 200
Subtotal produced by process with ID 13152: 200
Subtotal produced by process with ID 13153: 200
Subtotal produced by process with ID 13154: 200
Subtotal produced by process with ID 13155: 200
Subtotal produced by process with ID 13156: 200
Subtotal produced by process with ID 13157: 200
Subtotal produced by process with ID 13158: 200
Subtotal produced by process with ID 13159: 200
Subtotal produced by process with ID 13160: 200
Subtotal produced by process with ID 13161: 200
Subtotal produced by process with ID 13162: 200
Subtotal produced by process with ID 13163: 200
Subtotal produced by process with ID 13164: 200
Subtotal produced by process with ID 13165: 200
Subtotal produced by process with ID 13166: 200
Subtotal produced by process with ID 13167: 200
```

```
Subtotal produced by process with ID 13168: 200
Subtotal produced by process with ID 13169: 200
Subtotal produced by process with ID 13170: 200
Subtotal produced by process with ID 13171: 200
Subtotal produced by process with ID 13172: 200
Subtotal produced by process with ID 13173: 200
Subtotal produced by process with ID 13174: 200
Subtotal produced by process with ID 13175: 200
Subtotal produced by process with ID 13176: 200
Subtotal produced by process with ID 13177: 200
Subtotal produced by process with ID 13178: 200
Subtotal produced by process with ID 13179: 200
Subtotal produced by process with ID 13180: 200
Subtotal produced by process with ID 13181: 200
Subtotal produced by process with ID 13182: 200
Subtotal produced by process with ID 13183: 200
Subtotal produced by process with ID 13184: 200
Subtotal produced by process with ID 13185: 200
Subtotal produced by process with ID 13186: 200
Subtotal produced by process with ID 13187: 200
Subtotal produced by process with ID 13188: 200
Subtotal produced by process with ID 13189: 200
Subtotal produced by process with ID 13190: 200
Subtotal produced by process with ID 13191: 200
Subtotal produced by process with ID 13192: 200
Subtotal produced by process with ID 13193: 200
Subtotal produced by process with ID 13194: 200
Subtotal produced by process with ID 13195: 200
Subtotal produced by process with ID 13196: 200
Subtotal produced by process with ID 13197: 200
Subtotal produced by process with ID 13198: 200
Subtotal produced by process with ID 13199: 200
Subtotal produced by process with ID 13200: 200
Subtotal produced by process with ID 13201: 200
Subtotal produced by process with ID 13202: 200
Subtotal produced by process with ID 13203: 200
Subtotal produced by process with ID 13204: 200
Subtotal produced by process with ID 13205: 200
Subtotal produced by process with ID 13206: 200
Subtotal produced by process with ID 13207: 200
Subtotal produced by process with ID 13208: 200
Subtotal produced by process with ID 13209: 200
Subtotal produced by process with ID 13210: 200
Subtotal produced by process with ID 13211: 200
Subtotal produced by process with ID 13212: 200
Subtotal produced by process with ID 13213: 200
Subtotal produced by process with ID 13214: 200
Subtotal produced by process with ID 13215: 200
Subtotal produced by process with ID 13216: 200
Subtotal produced by process with ID 13217: 200
Subtotal produced by process with ID 13218: 200
Subtotal produced by process with ID 13219: 200
Subtotal produced by process with ID 13220: 200
```

```
Subtotal produced by process with ID 13221: 200
Subtotal produced by process with ID 13222: 200
Subtotal produced by process with ID 13223: 200
Subtotal produced by process with ID 13224: 200
Subtotal produced by process with ID 13225: 200
Subtotal produced by process with ID 13226: 200
Subtotal produced by process with ID 13227: 200
Subtotal produced by process with ID 13228: 200
Subtotal produced by process with ID 13229: 200
Subtotal produced by process with ID 13230: 200
Subtotal produced by process with ID 13231: 200
Subtotal produced by process with ID 13232: 200
Subtotal produced by process with ID 13233: 200
Subtotal produced by process with ID 13234: 200
Subtotal produced by process with ID 13235: 200
Subtotal produced by process with ID 13236: 200
Subtotal produced by process with ID 13237: 200
Subtotal produced by process with ID 13238: 200
Subtotal produced by process with ID 13239: 200
Subtotal produced by process with ID 13240: 200
Subtotal produced by process with ID 13241: 200
Subtotal produced by process with ID 13242: 200
Subtotal produced by process with ID 13243: 200
Subtotal produced by process with ID 13244: 200
Subtotal produced by process with ID 13245: 200
Subtotal produced by process with ID 13246: 200
Subtotal produced by process with ID 13247: 200
Subtotal produced by process with ID 13150: 200
Total: 20000
```

TEST 1 RESULTS
./pmmsThread test1MatrixA test1MatrixB 3 2 4

Subtotal produced by thread with ID 139785092052736: 134
Subtotal produced by thread with ID 139785100445440: 62
Subtotal produced by thread with ID 139785083660032: 206
Total: 402

TEST 2 RESULTS
./pmmsThread test2MatrixA test2MatrixB 20 11 40

Subtotal produced by thread with ID 140039828301568: 25420
Subtotal produced by thread with ID 140039819908864: 13940
Subtotal produced by thread with ID 140039811516160: 16400
Subtotal produced by thread with ID 140039677232896: 18860
Subtotal produced by thread with ID 140039803123456: 9840
Subtotal produced by thread with ID 140039794730752: 13940
Subtotal produced by thread with ID 140039786338048: 19680
Subtotal produced by thread with ID 140039777945344: 21320
Subtotal produced by thread with ID 140039769552640: 20500
Subtotal produced by thread with ID 140039761159936: 5740
Subtotal produced by thread with ID 140039752767232: 7380
Subtotal produced by thread with ID 140039744374528: 20500
Subtotal produced by thread with ID 140039735981824: 17220
Subtotal produced by thread with ID 140039727589120: 22960
Subtotal produced by thread with ID 140039719196416: 20500
Subtotal produced by thread with ID 140039710803712: 14760
Subtotal produced by thread with ID 140039702411008: 18860
Subtotal produced by thread with ID 140039694018304: 20500
Subtotal produced by thread with ID 140039685625600: 13940
Subtotal produced by thread with ID 140039828301568: 18860
Total: 341120

TEST 3 RESULTS
./pmmsThread test3MatrixA test3MatrixB 93 21 6

Subtotal produced by thread with ID 139855865177856: 982
Subtotal produced by thread with ID 139855873570560: 1430
Subtotal produced by thread with ID 139855856785152: 445
Subtotal produced by thread with ID 139855109834496: 442
Subtotal produced by thread with ID 139855865177856: 1128
Subtotal produced by thread with ID 139855839999744: 1040
Subtotal produced by thread with ID 139855823214336: 1213
Subtotal produced by thread with ID 139855831607040: 1613
Subtotal produced by thread with ID 139855814821632: 1887
Subtotal produced by thread with ID 139855798036224: 1082
Subtotal produced by thread with ID 139855806428928: 1311
Subtotal produced by thread with ID 139855789643520: 2009
Subtotal produced by thread with ID 139855781250816: 1418

```
Subtotal produced by thread with ID 139855772858112: 979
Subtotal produced by thread with ID 139855764465408: 2433
Subtotal produced by thread with ID 139855747680000: 1271
Subtotal produced by thread with ID 139855756072704: 2244
Subtotal produced by thread with ID 139855739287296: 1805
Subtotal produced by thread with ID 139855722501888: 884
Subtotal produced by thread with ID 139855730894592: 2442
Subtotal produced by thread with ID 139855714109184: 823
Subtotal produced by thread with ID 139855705716480: 2372
Subtotal produced by thread with ID 139855697323776: 2381
Subtotal produced by thread with ID 139855680538368: 509
Subtotal produced by thread with ID 139855688931072: 1348
Subtotal produced by thread with ID 139855672145664: 1857
Subtotal produced by thread with ID 139855663752960: 1253
Subtotal produced by thread with ID 139855655360256: 2195
Subtotal produced by thread with ID 139855646967552: 433
Subtotal produced by thread with ID 139855638574848: 1137
Subtotal produced by thread with ID 139855630182144: 1619
Subtotal produced by thread with ID 139855621789440: 918
Subtotal produced by thread with ID 139855613396736: 1561
Subtotal produced by thread with ID 139855605004032: 1412
Subtotal produced by thread with ID 139855596611328: 1488
Subtotal produced by thread with ID 139855588218624: 1442
Subtotal produced by thread with ID 139855579825920: 1409
Subtotal produced by thread with ID 139855571433216: 1338
Subtotal produced by thread with ID 139855563040512: 1509
Subtotal produced by thread with ID 139855554647808: 698
Subtotal produced by thread with ID 139855546255104: 1570
Subtotal produced by thread with ID 139855537862400: 1506
Subtotal produced by thread with ID 139855529469696: 1729
Subtotal produced by thread with ID 139855521076992: 503
Subtotal produced by thread with ID 139855512684288: 2015
Subtotal produced by thread with ID 139855504291584: 1680
Subtotal produced by thread with ID 139855495898880: 826
Subtotal produced by thread with ID 139855487506176: 863
Subtotal produced by thread with ID 139855479113472: 1738
Subtotal produced by thread with ID 139855470720768: 186
Subtotal produced by thread with ID 139855462328064: 1805
Subtotal produced by thread with ID 139855453935360: 2009
Subtotal produced by thread with ID 139855445542656: 1695
Subtotal produced by thread with ID 139855437149952: 1439
Subtotal produced by thread with ID 139855428757248: 1604
Subtotal produced by thread with ID 139855420364544: 308
Subtotal produced by thread with ID 139855411971840: 1168
Subtotal produced by thread with ID 139855403579136: 2067
Subtotal produced by thread with ID 139855395186432: 1451
Subtotal produced by thread with ID 139855386793728: 442
Subtotal produced by thread with ID 139855378401024: 692
Subtotal produced by thread with ID 139855370008320: 1793
Subtotal produced by thread with ID 139855361615616: 2177
Subtotal produced by thread with ID 139855353222912: 2061
Subtotal produced by thread with ID 139855344830208: 1872
Subtotal produced by thread with ID 139855336437504: 1284
```

```
Subtotal produced by thread with ID 139855328044800: 1497
Subtotal produced by thread with ID 139855319652096: 1924
Subtotal produced by thread with ID 139855311259392: 857
Subtotal produced by thread with ID 139855302866688: 1701
Subtotal produced by thread with ID 139855294473984: 1732
Subtotal produced by thread with ID 139855286081280: 1311
Subtotal produced by thread with ID 139855277688576: 826
Subtotal produced by thread with ID 139855269295872: 1149
Subtotal produced by thread with ID 139855260903168: 1396
Subtotal produced by thread with ID 139855252510464: 738
Subtotal produced by thread with ID 139855244117760: 2183
Subtotal produced by thread with ID 139855235725056: 1430
Subtotal produced by thread with ID 139855227332352: 1619
Subtotal produced by thread with ID 139855218939648: 189
Subtotal produced by thread with ID 139855210546944: 1543
Subtotal produced by thread with ID 139855202154240: 939
Subtotal produced by thread with ID 139855193761536: 1052
Subtotal produced by thread with ID 139855185368832: 1613
Subtotal produced by thread with ID 139855176976128: 2253
Subtotal produced by thread with ID 139855168583424: 951
Subtotal produced by thread with ID 139855160190720: 1796
Subtotal produced by thread with ID 139855151798016: 1195
Subtotal produced by thread with ID 139855143405312: 314
Subtotal produced by thread with ID 139855135012608: 1869
Subtotal produced by thread with ID 139855126619904: 2049
Subtotal produced by thread with ID 139855118227200: 1808
Subtotal produced by thread with ID 139855848392448: 1634
Total: 129811
```

TEST 4 RESULTS

```
./pmmsThread test4MatrixA test4MatrixB 100 100 100

Subtotal produced by thread with ID 139638019942144: 200
Subtotal produced by thread with ID 139638028334848: 200
Subtotal produced by thread with ID 139638003156736: 200
Subtotal produced by thread with ID 139637214242560: 200
Subtotal produced by thread with ID 139637986371328: 200
Subtotal produced by thread with ID 139637994764032: 200
Subtotal produced by thread with ID 139637977978624: 200
Subtotal produced by thread with ID 139638019942144: 200
Subtotal produced by thread with ID 139637969585920: 200
Subtotal produced by thread with ID 139637961193216: 200
Subtotal produced by thread with ID 139637944407808: 200
Subtotal produced by thread with ID 139637952800512: 200
Subtotal produced by thread with ID 139637936015104: 200
Subtotal produced by thread with ID 139637927622400: 200
Subtotal produced by thread with ID 139637919229696: 200
Subtotal produced by thread with ID 139637910836992: 200
Subtotal produced by thread with ID 139637902444288: 200
Subtotal produced by thread with ID 139637894051584: 200
Subtotal produced by thread with ID 139637877266176: 200
```

```
Subtotal produced by thread with ID 139637885658880: 200
Subtotal produced by thread with ID 139637868873472: 200
Subtotal produced by thread with ID 139637860480768: 200
Subtotal produced by thread with ID 139637852088064: 200
Subtotal produced by thread with ID 139637843695360: 200
Subtotal produced by thread with ID 139637835302656: 200
Subtotal produced by thread with ID 139637826909952: 200
Subtotal produced by thread with ID 139637818517248: 200
Subtotal produced by thread with ID 139637810124544: 200
Subtotal produced by thread with ID 139637801731840: 200
Subtotal produced by thread with ID 139637793339136: 200
Subtotal produced by thread with ID 139637784946432: 200
Subtotal produced by thread with ID 139637776553728: 200
Subtotal produced by thread with ID 139637768161024: 200
Subtotal produced by thread with ID 139637759768320: 200
Subtotal produced by thread with ID 139637751375616: 200
Subtotal produced by thread with ID 139637742982912: 200
Subtotal produced by thread with ID 139637734590208: 200
Subtotal produced by thread with ID 139637726197504: 200
Subtotal produced by thread with ID 139637717804800: 200
Subtotal produced by thread with ID 139637709412096: 200
Subtotal produced by thread with ID 139637701019392: 200
Subtotal produced by thread with ID 139637692626688: 200
Subtotal produced by thread with ID 139637684233984: 200
Subtotal produced by thread with ID 139637675841280: 200
Subtotal produced by thread with ID 139637667448576: 200
Subtotal produced by thread with ID 139637659055872: 200
Subtotal produced by thread with ID 139637650663168: 200
Subtotal produced by thread with ID 139637642270464: 200
Subtotal produced by thread with ID 139637633877760: 200
Subtotal produced by thread with ID 139637625485056: 200
Subtotal produced by thread with ID 139637617092352: 200
Subtotal produced by thread with ID 139637608699648: 200
Subtotal produced by thread with ID 139637600306944: 200
Subtotal produced by thread with ID 139637591914240: 200
Subtotal produced by thread with ID 139637583521536: 200
Subtotal produced by thread with ID 139637575128832: 200
Subtotal produced by thread with ID 139637566736128: 200
Subtotal produced by thread with ID 139637558343424: 200
Subtotal produced by thread with ID 139637549950720: 200
Subtotal produced by thread with ID 139637541558016: 200
Subtotal produced by thread with ID 139637533165312: 200
Subtotal produced by thread with ID 139637524772608: 200
Subtotal produced by thread with ID 139637516379904: 200
Subtotal produced by thread with ID 139637507987200: 200
Subtotal produced by thread with ID 139637499594496: 200
Subtotal produced by thread with ID 139637491201792: 200
Subtotal produced by thread with ID 139637482809088: 200
Subtotal produced by thread with ID 139637474416384: 200
Subtotal produced by thread with ID 139637457630976: 200
Subtotal produced by thread with ID 139637449238272: 200
Subtotal produced by thread with ID 139637440845568: 200
Subtotal produced by thread with ID 139637466023680: 200
```

```
Subtotal produced by thread with ID 139637432452864: 200
Subtotal produced by thread with ID 139637424060160: 200
Subtotal produced by thread with ID 139637415667456: 200
Subtotal produced by thread with ID 139637407274752: 200
Subtotal produced by thread with ID 139637398882048: 200
Subtotal produced by thread with ID 139637390489344: 200
Subtotal produced by thread with ID 139637382096640: 200
Subtotal produced by thread with ID 139637373703936: 200
Subtotal produced by thread with ID 139637365311232: 200
Subtotal produced by thread with ID 139637356918528: 200
Subtotal produced by thread with ID 139637348525824: 200
Subtotal produced by thread with ID 139637340133120: 200
Subtotal produced by thread with ID 139637256206080: 200
Subtotal produced by thread with ID 139637331740416: 200
Subtotal produced by thread with ID 139637264598784: 200
Subtotal produced by thread with ID 139637323347712: 200
Subtotal produced by thread with ID 139637272991488: 200
Subtotal produced by thread with ID 139637289776896: 200
Subtotal produced by thread with ID 139637298169600: 200
Subtotal produced by thread with ID 139637306562304: 200
Subtotal produced by thread with ID 139637314955008: 200
Subtotal produced by thread with ID 139637281384192: 200
Subtotal produced by thread with ID 139637247813376: 200
Subtotal produced by thread with ID 139637239420672: 200
Subtotal produced by thread with ID 139637231027968: 200
Subtotal produced by thread with ID 139638011549440: 200
Subtotal produced by thread with ID 139637205849856: 200
Subtotal produced by thread with ID 139637222635264: 200
Total: 20000
```