Name: Mayesha Bintha Mizan
ID-811281302

The goal of this project is to design and implement a matrix multiplication program using both multi-process and multi-threaded approaches.

Parts 1 and 2: Multi-Process and Multi-Threaded Matrix Multiplication

In the **multi-process model,** I divided the matrix multiplication workload across multipl child processes. The parent process creates the specified number of child processes **(numProc)** to distribute the workload evenly. The number of rows assigned to each process is calculated as **rows_per_proc = m / numProc**, where m is the number of rows in matrix A. If the number of rows is not evenly divisible by the number of processes, the remaining rows are assigned to the last process.

The main logic of each child process is implemented in the **child_process_core() function**. The parent process uses **the fork() system call** to create numProc child processes, each of which executes the **child_process_core() function**. This function handles matrix multiplication for the rows assigned to that process by reading the relevant rows from matrix A, multiplying them with matrix B, and computing the result for the corresponding section of matrix C.

I implemented **pipes** as **the IPC mechanism** to gather results from the child processes. Each child process writes its computed results into a dedicated pipe, and the parent process reads from these pipes to collect and assemble the final result matrix. This method ensures that the parent can retrieve the results from all child processes once they have completed their portion of the matrix multiplication.

In the **multi-threaded model,** I used a similar approach to the multi-process model, dividing the matrix multiplication task across multiple threads. Each thread is responsible for computing a portion of the result matrix, with the number of rows assigned to each thread calculated as **rows_per_thread = m / numThread,** where m is the number of rows in matrix A and **numThread** is the number of threads. If the number of rows is not evenly divisible by the number of threads, the last thread handles any remaining rows.

Each thread executes the **thread_core() function**, which performs matrix multiplication for the assigned rows. I used the **tlist_t** structure to store job-related information for each thread, such as **the thread ID (tid), the starting row (row_start), and the ending row (row_end**). Threads write their computed results directly into **the global result matrix C_multi_thread,** which is accessible to all threads since they share the same memory space.

Thread management is handled using **pthread_create()** to launch the threads, and **pthread_join()** ensures that all threads complete their assigned tasks. If a thread crashes, it is either restarted or its task is reassigned to another thread. Once a thread finishes computing its assigned rows, it exits using **pthread_exit().** The program ensures that the matrix multiplication is completed by monitoring the status of each thread. If any rows remain unprocessed due to thread failures, the main thread creates new threads to complete the remaining tasks.

The child processes simulate crashes based on **a user-specified crash rate (0-30).** In both the multi-process and multi-threaded models, I used **a circular queue** to track the job status for processes **(plist_t)** and threads (**tlist_t).** This queue monitors the progress of each job and ensures that any failures are handled by restarting the process/thread or reassigning the task to another. This design ensures robust fault tolerance, guaranteeing that the matrix multiplication finishes even in the presence of failures.

Part 3: Performance Evaluation

To evaluate the scalability and performance of the different matrix multiplication models (single-process, multi-process, and multi-threaded), I tested matrices of varying dimensions (m, n, p) to simulate small, medium, and large datasets. This approach allowed to observe how the models handled different workloads and how matrix size influenced their performance.

The matrix sizes tested included:

- **Small matrices**: A (4x3), B (3x4)
- **Medium matrices**: A (10x20), B (20x30)
- **Large matrices**: A (10x1000), B (1000x1000)
- **Very large matrices**: A (20x2000), B (2000x2000)

Test 1: Matrix Size A (4x3), B (3x4)

- Serial multiplication took: 0.000000 seconds.
- Multi-process multiplication took: 0.000608 seconds.
- Multi-threaded multiplication took: 0.000447 seconds.

For this small matrix size, both multi-process and multi-threaded models completed the task almost instantaneously. The multi-threaded model was slightly faster. The results were again identical across all models, verifying their accuracy.

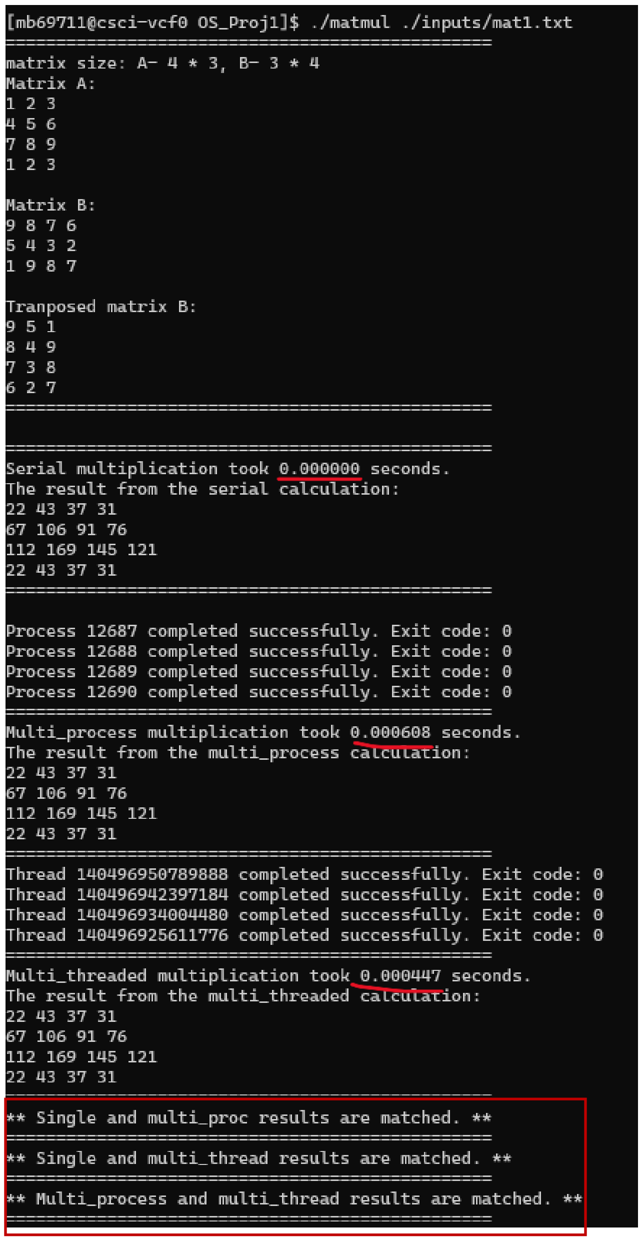
Test 2: Matrix Size A (10x20), B (20x30)

- Serial multiplication took: 0.000014 seconds.
- Multi-process multiplication took: 0.001260 seconds.
- Multi-threaded multiplication took: 0.00775 seconds.

In the medium-sized matrix test, the multi-threaded model again outperformed the multi-process model. The overhead of process management and IPC became more evident as matrix size increased. The multi-threaded model took less time to complete due to efficient memory sharing between threads.

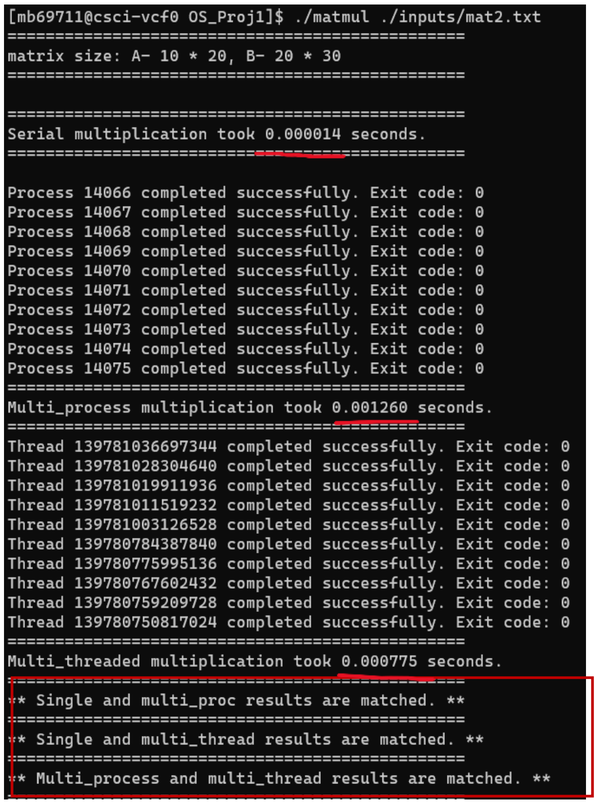
Name: Mayesha Bintha Mizan
ID-811281302

Test-1



```
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat1.txt
==============================================
matrix size: A- 4 * 3, B- 3 * 4
Matrix A:
1 2 3
4 5 6
7 8 9
1 2 3

Matrix B:
9 8 7 6
5 4 3 2
1 9 8 7

Tranposed matrix B:
9 5 1
8 4 9
7 3 8
6 2 7
==============================================

==============================================
Serial multiplication took 0.000000 seconds.
The result from the serial calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
==============================================

Process 12687 completed successfully. Exit code: 0
Process 12688 completed successfully. Exit code: 0
Process 12689 completed successfully. Exit code: 0
Process 12690 completed successfully. Exit code: 0
==============================================
Multi_process multiplication took 0.000608 seconds.
The result from the multi_process calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
==============================================
Thread 140496950789888 completed successfully. Exit code: 0
Thread 140496942397184 completed successfully. Exit code: 0
Thread 140496934004480 completed successfully. Exit code: 0
Thread 140496925611776 completed successfully. Exit code: 0
==============================================
Multi_threaded multiplication took 0.000447 seconds.
The result from the multi_threaded calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
==============================================
** Single and multi_proc results are matched. **
==============================================
** Single and multi_thread results are matched. **
==============================================
** Multi_process and multi_thread results are matched. **
==============================================
```

Test-2



```
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat2.txt
==============================================
matrix size: A- 10 * 20, B- 20 * 30
==============================================

==============================================
Serial multiplication took 0.000014 seconds.
==============================================

Process 14066 completed successfully. Exit code: 0
Process 14067 completed successfully. Exit code: 0
Process 14068 completed successfully. Exit code: 0
Process 14069 completed successfully. Exit code: 0
Process 14070 completed successfully. Exit code: 0
Process 14071 completed successfully. Exit code: 0
Process 14072 completed successfully. Exit code: 0
Process 14073 completed successfully. Exit code: 0
Process 14074 completed successfully. Exit code: 0
Process 14075 completed successfully. Exit code: 0
==============================================
Multi_process multiplication took 0.001260 seconds.
==============================================
Thread 139781036697344 completed successfully. Exit code: 0
Thread 139781028304640 completed successfully. Exit code: 0
Thread 139781019911936 completed successfully. Exit code: 0
Thread 139781011519232 completed successfully. Exit code: 0
Thread 139781003126528 completed successfully. Exit code: 0
Thread 139780784387840 completed successfully. Exit code: 0
Thread 139780775995136 completed successfully. Exit code: 0
Thread 139780767602432 completed successfully. Exit code: 0
Thread 139780759209728 completed successfully. Exit code: 0
Thread 139780750817024 completed successfully. Exit code: 0
==============================================
Multi_threaded multiplication took 0.000775 seconds.
==============================================
** Single and multi_proc results are matched. **
==============================================
** Single and multi_thread results are matched. **
==============================================
** Multi_process and multi_thread results are matched. **
```

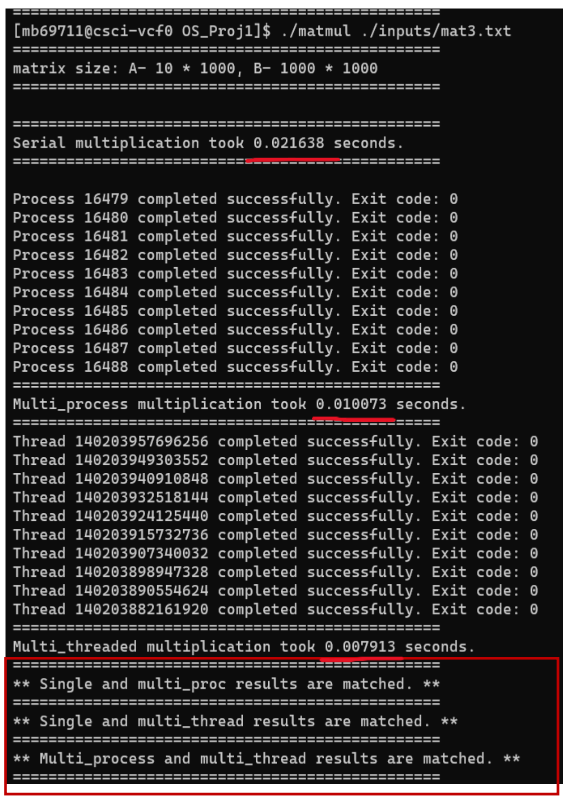
Test 3: Large Matrix Size A (10x1000), B (1000x1000)

- Serial multiplication: 0.021638 seconds
- Multi-process multiplication: 0.010073 seconds
- Multi-threaded multiplication: 0.007913 seconds

Name: Mayesha Bintha Mizan
ID-811281302

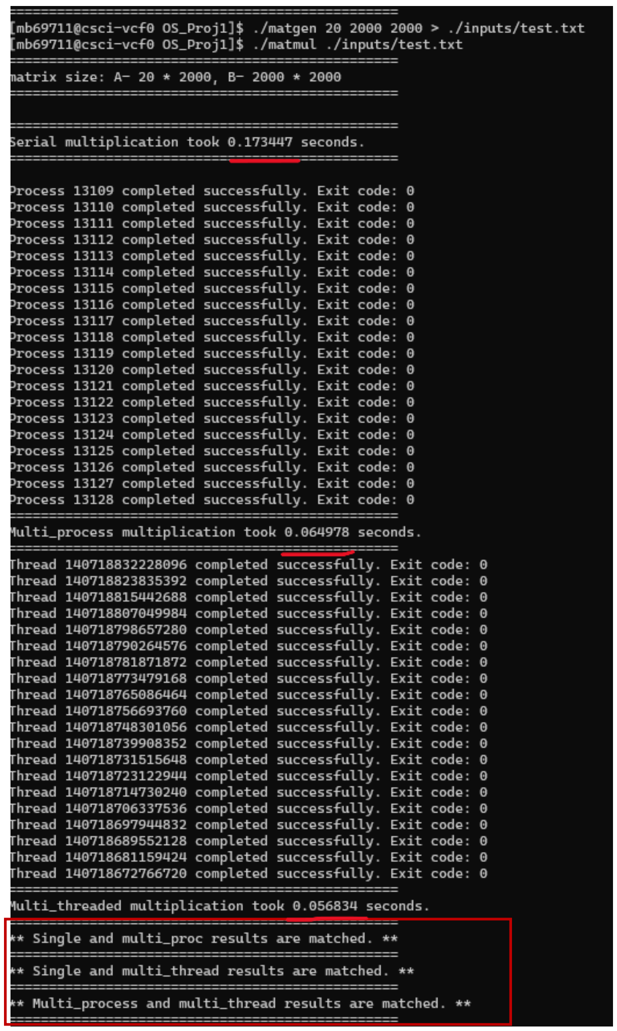
The multi-threaded model completed the task faster than the multi-process model for this large matrix. The serial model, as expected, took the longest time. The **multi-threaded model** showed a clear advantage here, completing the matrix multiplication efficiently due to shared memory.

Test-3



```
================================================
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat3.txt
================================================
matrix size: A- 10 * 1000, B- 1000 * 1000
================================================

================================================
Serial multiplication took 0.021638 seconds.
================================================

Process 16479 completed successfully. Exit code: 0
Process 16480 completed successfully. Exit code: 0
Process 16481 completed successfully. Exit code: 0
Process 16482 completed successfully. Exit code: 0
Process 16483 completed successfully. Exit code: 0
Process 16484 completed successfully. Exit code: 0
Process 16485 completed successfully. Exit code: 0
Process 16486 completed successfully. Exit code: 0
Process 16487 completed successfully. Exit code: 0
Process 16488 completed successfully. Exit code: 0
================================================
Multi_process multiplication took 0.010073 seconds.
================================================
Thread 140203957696256 completed successfully. Exit code: 0
Thread 140203949303552 completed successfully. Exit code: 0
Thread 140203940910848 completed successfully. Exit code: 0
Thread 140203932518144 completed successfully. Exit code: 0
Thread 140203924125440 completed successfully. Exit code: 0
Thread 140203915732736 completed successfully. Exit code: 0
Thread 140203907340032 completed successfully. Exit code: 0
Thread 140203898947328 completed successfully. Exit code: 0
Thread 140203890554624 completed successfully. Exit code: 0
Thread 140203882161920 completed successfully. Exit code: 0
================================================
Multi_threaded multiplication took 0.007913 seconds.
================================================
** Single and multi_proc results are matched. **
================================================
** Single and multi_thread results are matched. **
================================================
** Multi_process and multi_thread results are matched. **
================================================
```

Test-4



```
================================================
[mb69711@csci-vcf0 OS_Proj1]$ ./matgen 20 2000 2000 > ./inputs/test.txt
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/test.txt
================================================
atrix size: A- 20 * 2000, B- 2000 * 2000
================================================

================================================
erial multiplication took 0.173447 seconds.
================================================

rocess 13109 completed successfully. Exit code: 0
rocess 13110 completed successfully. Exit code: 0
rocess 13111 completed successfully. Exit code: 0
rocess 13112 completed successfully. Exit code: 0
rocess 13113 completed successfully. Exit code: 0
rocess 13114 completed successfully. Exit code: 0
rocess 13115 completed successfully. Exit code: 0
rocess 13116 completed successfully. Exit code: 0
rocess 13117 completed successfully. Exit code: 0
rocess 13118 completed successfully. Exit code: 0
rocess 13119 completed successfully. Exit code: 0
rocess 13120 completed successfully. Exit code: 0
rocess 13121 completed successfully. Exit code: 0
rocess 13122 completed successfully. Exit code: 0
rocess 13123 completed successfully. Exit code: 0
rocess 13124 completed successfully. Exit code: 0
rocess 13125 completed successfully. Exit code: 0
rocess 13126 completed successfully. Exit code: 0
rocess 13127 completed successfully. Exit code: 0
rocess 13128 completed successfully. Exit code: 0
================================================
ulti_process multiplication took 0.064978 seconds.
================================================
hread 140718832228096 completed successfully. Exit code: 0
hread 140718823835392 completed successfully. Exit code: 0
hread 140718815442688 completed successfully. Exit code: 0
hread 140718807049984 completed successfully. Exit code: 0
hread 140718798657280 completed successfully. Exit code: 0
hread 140718790264576 completed successfully. Exit code: 0
hread 140718781871872 completed successfully. Exit code: 0
hread 140718773479168 completed successfully. Exit code: 0
hread 140718765086464 completed successfully. Exit code: 0
hread 140718756693760 completed successfully. Exit code: 0
hread 140718748301056 completed successfully. Exit code: 0
hread 140718739908352 completed successfully. Exit code: 0
hread 140718731515648 completed successfully. Exit code: 0
hread 140718723122944 completed successfully. Exit code: 0
hread 140718714730240 completed successfully. Exit code: 0
hread 140718706337536 completed successfully. Exit code: 0
hread 140718697944832 completed successfully. Exit code: 0
hread 140718689552128 completed successfully. Exit code: 0
hread 140718681159424 completed successfully. Exit code: 0
hread 140718672766720 completed successfully. Exit code: 0
================================================
ulti_threaded multiplication took 0.056834 seconds.
================================================
* Single and multi_proc results are matched. **
================================================
* Single and multi_thread results are matched. **
================================================
* Multi_process and multi_thread results are matched. **
================================================
```

Test 4: Matrix Size A (20x2000), B (2000x2000)

- Serial multiplication took: 0.173447 seconds.
- Multi-process multiplication took: 0.064978 seconds.
- Multi-threaded multiplication took: 0.056834 seconds.

For this large matrix, the multi-threaded model significantly outperformed both the serial and multi-process models due to its ability to handle large data more efficiently. The serial computation took almost three times longer than the parallel models.

The results clearly showed that the **multi-threaded approach consistently outperformed the multi-process approach**, particularly for medium and large matrix sizes. With small matrices, the performance difference between the two parallel models was minimal, but the multi-threaded model still had a slight advantage due to its lower overhead.

Unexpected Results: Sometimes the multi-process model slightly outperformed the multi-threaded model. While this was not typical of the other test cases, this variance could be explained by system-level resource contention, such as CPU scheduling, thread management, or cache behavior, which may have favored processes over threads at the time of testing. Further tests with different system conditions might provide a clearer understanding of these occasional discrepancies.

Part 4: Recovery from the crash

In both the multi-process and multi-threaded matrix multiplication models, I have implemented a robust crash handling mechanism using **a circular queue**. This ensures that matrix multiplication tasks are completed even when processes or threads fail during execution. The key to the crash recovery strategy is managing retries efficiently, up to a maximum of three attempts, to maintain the correctness and performance of the matrix multiplication task.

For both, the parent process or main thread pulls job information from **a circular queue**.This is how crash handling will work,

- pull an element from the queue
- wait for the child process result
- if child process normally terminated, calculate the result
- else create a new child process with the info from that element and add it to queue again

This will continue until the queue is empty.

In the multi-process model, the parent process uses **waitpid()** to wait for child processes to complete, while in the multi-threaded model, the main thread uses **pthread_join()** to monitor the status of worker threads. Both models detect whether the process or thread completes successfully or crashes.

If the process or thread finishes successfully, the parent process or main thread collects the computed results:

- **Multi-Process**: The parent reads the results from a pipe, where the child process has written its portion of the matrix result.
- **Multi-Threaded**: Since threads share memory, the results are directly written into the global matrix.

This marks the successful completion of the job, and no further action is required for that process or thread.

If a process or thread crashes, it is detected via **WIFSIGNALED(status)** in the process model or the abnormal exit status from **pthread_join()** in the thread model. The system logs the crash, retrieves the corresponding job from the queue, and retries the task by creating a new process or thread.

The new process or thread is added back to the circular queue for further monitoring.

The system allows up to three retry attempts for each failed process or thread. If the task fails after three retries, it is abandoned, and no further attempts are made to complete that job. This prevents the system from getting stuck in an endless loop of retries and ensures efficient resource management.

The crash handling loop continues until the circular queue is empty. This guarantees that all jobs are either completed successfully or stopped after reaching the retry limit. The system ensures that the matrix multiplication task finishes, even if certain parts fail multiple times.

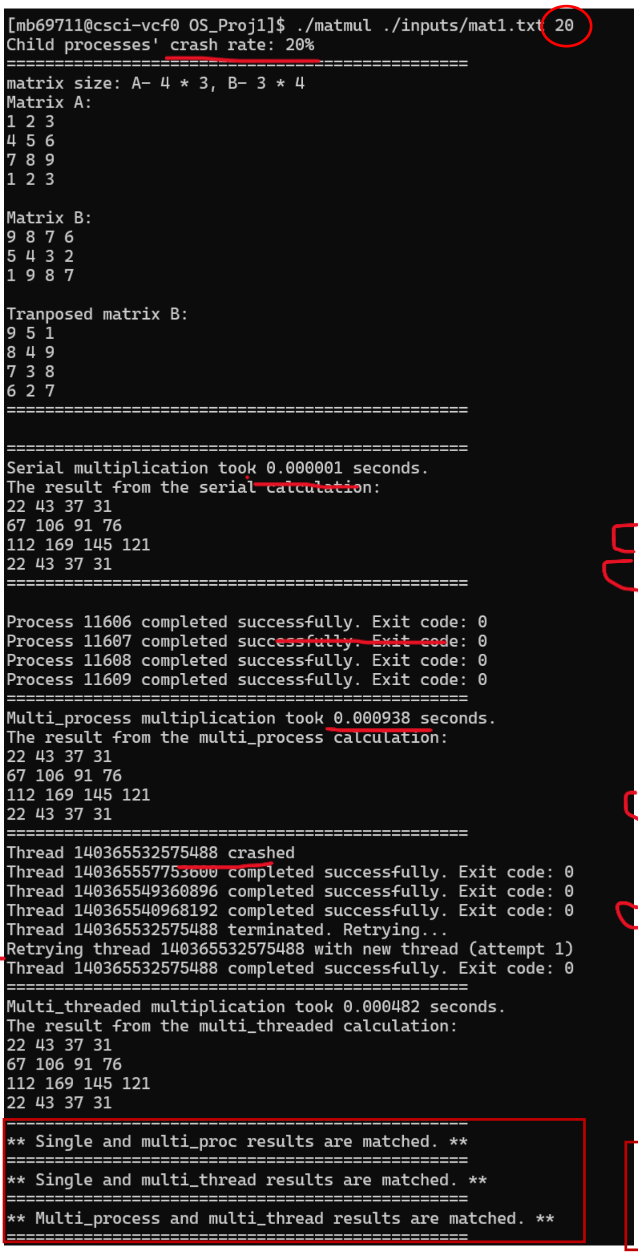
I tested them under various conditions to test their crash handling capabilities. The following observations were made:

Test 5 and 6: In the mat1 input file I apply 20% & 30% crash rate. Child processes and threads are assigned their tasks, but due to the crash simulation, some processes and threads terminate abnormally. When a crash is detected (e.g., "Process 11766 crashed" or " Thread 140681520961280 crashed"), the program immediately retries the failed process or thread. The retry is done by creating a new process or thread. The retry mechanism attempts to complete the task up to three times if a crash persists, as seen in messages like " Retrying process 11766 with new process 11770 (attempt 1)" and "Retrying thread with new thread (attempt 1)." After the retries, the child processes and threads complete successfully, as indicated by "Process completed successfully. Exit code: 0" and similar messages for threads. The matching results confirm that despite the crashes and retries, the calculations were accurate and consistent across all models.

The result may vary each time as for first time the process can take longer time to prepare but after that it run very first. The retry mechanism ensures that crashes do not permanently halt the computation, though a higher crash rate leads to more retries, which could slightly increase the total execution time .

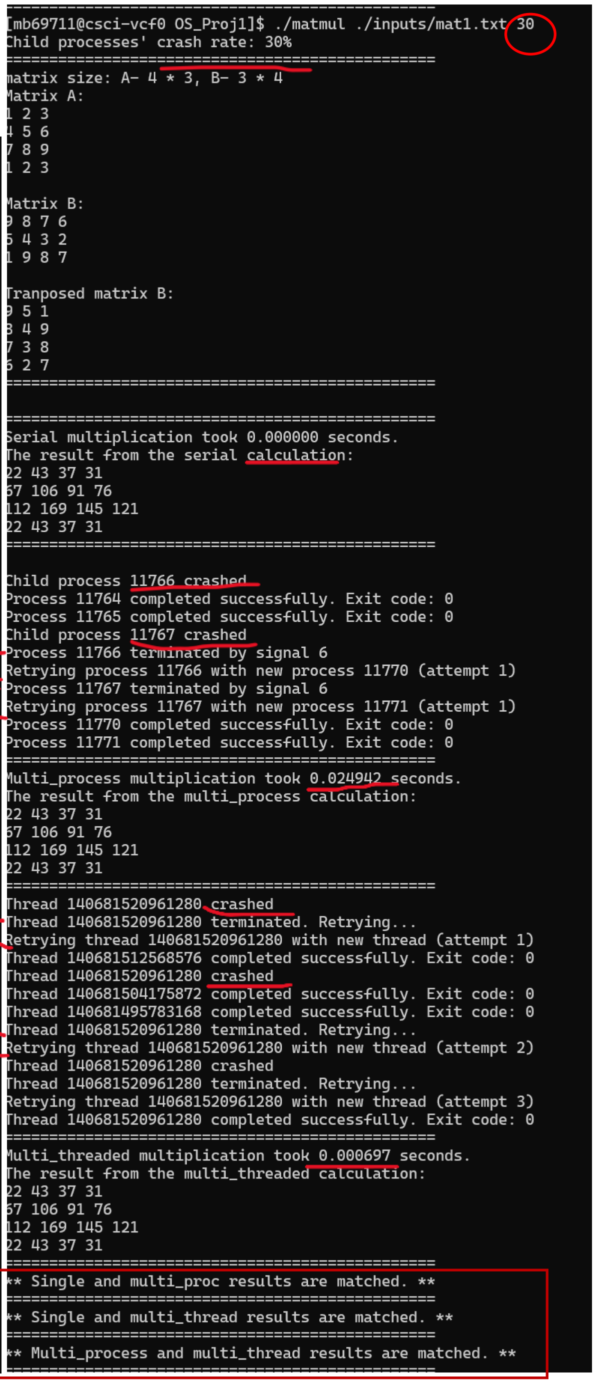
Name: Mayesha Bintha Mizan
ID-811281302

Test 5



```
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat1.txt 20
Child processes' crash rate: 20%
================================================
matrix size: A- 4 * 3, B- 3 * 4
Matrix A:
1 2 3
4 5 6
7 8 9
1 2 3

Matrix B:
9 8 7 6
5 4 3 2
1 9 8 7

Tranposed matrix B:
9 5 1
8 4 9
7 3 8
6 2 7
================================================

================================================
Serial multiplication took 0.000001 seconds.
The result from the serial calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================

Process 11606 completed successfully. Exit code: 0
Process 11607 completed successfully. Exit code: 0
Process 11608 completed successfully. Exit code: 0
Process 11609 completed successfully. Exit code: 0
================================================
Multi_process multiplication took 0.000938 seconds.
The result from the multi_process calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================
Thread 140365532575488 crashed
Thread 140365557753600 completed successfully. Exit code: 0
Thread 140365549360896 completed successfully. Exit code: 0
Thread 140365540968192 completed successfully. Exit code: 0
Thread 140365532575488 terminated. Retrying...
Retrying thread 140365532575488 with new thread (attempt 1)
Thread 140365532575488 completed successfully. Exit code: 0
================================================
Multi_threaded multiplication took 0.000482 seconds.
The result from the multi_threaded calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================
** Single and multi_proc results are matched. **
================================================
** Single and multi_thread results are matched. **
================================================
** Multi_process and multi_thread results are matched. **
================================================
```

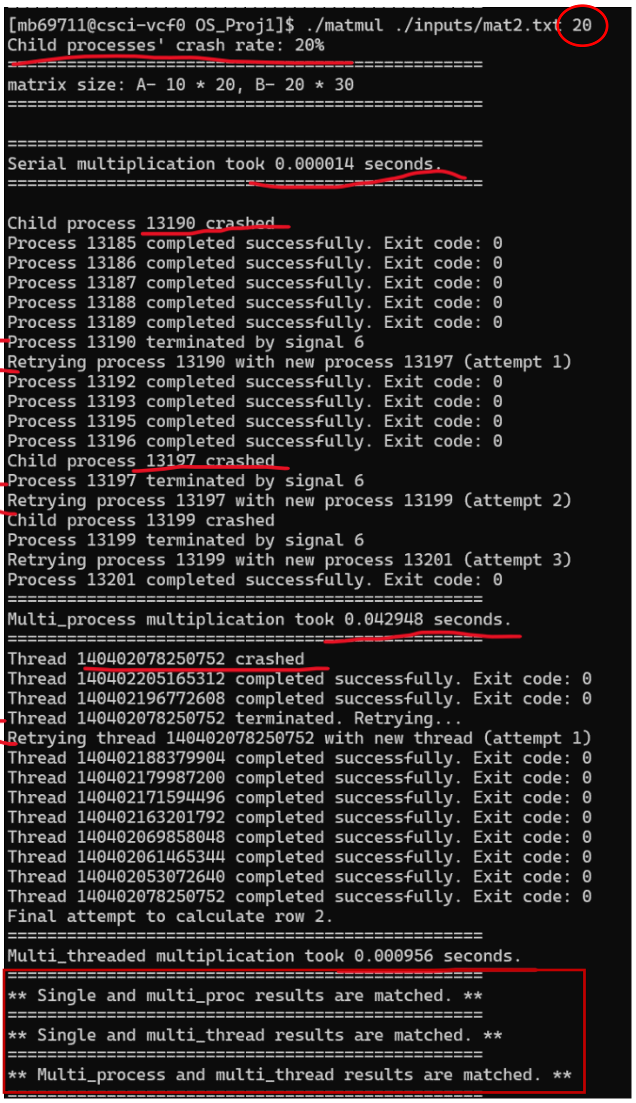
Test 6



```
================================================
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat1.txt 30
Child processes' crash rate: 30%
================================================
matrix size: A- 4 * 3, B- 3 * 4
Matrix A:
1 2 3
4 5 6
7 8 9
1 2 3

Matrix B:
9 8 7 6
5 4 3 2
1 9 8 7

Tranposed matrix B:
9 5 1
8 4 9
7 3 8
6 2 7
================================================

================================================
Serial multiplication took 0.000000 seconds.
The result from the serial calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================

Child process 11766 crashed
Process 11764 completed successfully. Exit code: 0
Process 11765 completed successfully. Exit code: 0
Child process 11767 crashed
Process 11766 terminated by signal 6
Retrying process 11766 with new process 11770 (attempt 1)
Process 11767 terminated by signal 6
Retrying process 11767 with new process 11771 (attempt 1)
Process 11770 completed successfully. Exit code: 0
Process 11771 completed successfully. Exit code: 0
================================================
Multi_process multiplication took 0.024942 seconds.
The result from the multi_process calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================
Thread 140681520961280 crashed
Thread 140681520961280 terminated. Retrying...
Retrying thread 140681520961280 with new thread (attempt 1)
Thread 140681512568576 completed successfully. Exit code: 0
Thread 140681520961280 crashed
Thread 140681504175872 completed successfully. Exit code: 0
Thread 140681495783168 completed successfully. Exit code: 0
Thread 140681520961280 terminated. Retrying...
Retrying thread 140681520961280 with new thread (attempt 2)
Thread 140681520961280 crashed
Thread 140681520961280 terminated. Retrying...
Retrying thread 140681520961280 with new thread (attempt 3)
Thread 140681520961280 completed successfully. Exit code: 0
================================================
Multi_threaded multiplication took 0.000697 seconds.
The result from the multi_threaded calculation:
22 43 37 31
67 106 91 76
112 169 145 121
22 43 37 31
================================================
** Single and multi_proc results are matched. **
================================================
** Single and multi_thread results are matched. **
================================================
** Multi_process and multi_thread results are matched. **
================================================
```

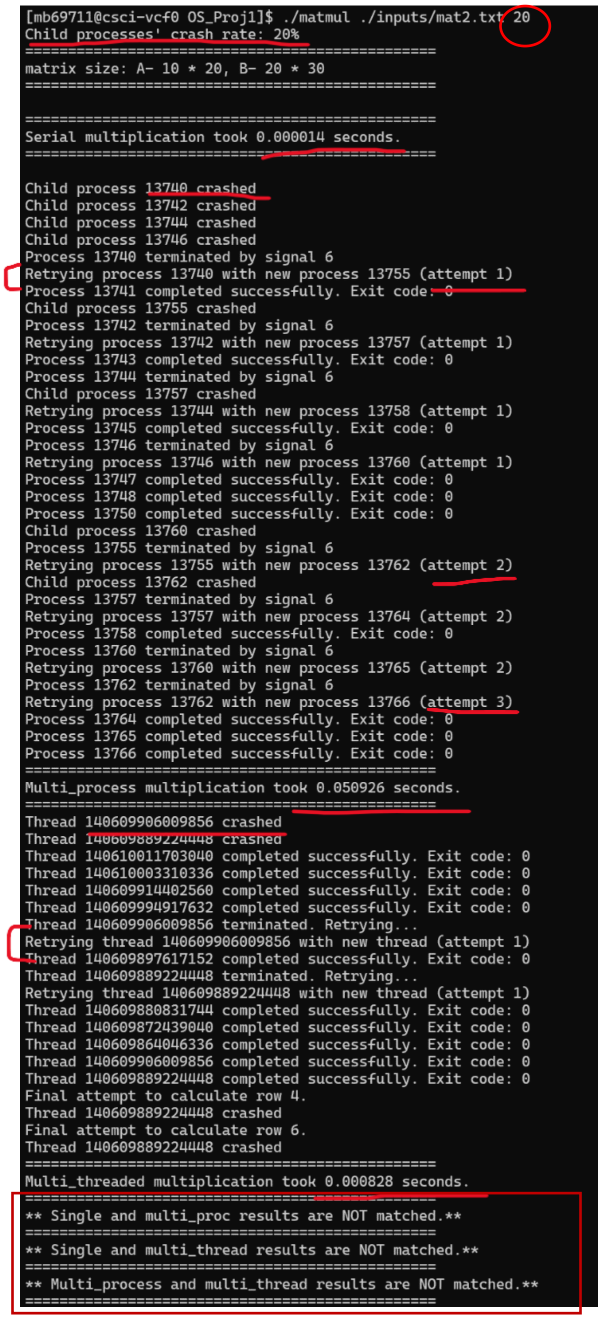
Name: Mayesha Bintha Mizan
ID-811281302

Test 7



```
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat2.txt 20
Child processes' crash rate: 20%
==============================================
matrix size: A- 10 * 20, B- 20 * 30
==============================================

==============================================
Serial multiplication took 0.000014 seconds.
==============================================

Child process 13190 crashed
Process 13185 completed successfully. Exit code: 0
Process 13186 completed successfully. Exit code: 0
Process 13187 completed successfully. Exit code: 0
Process 13188 completed successfully. Exit code: 0
Process 13189 completed successfully. Exit code: 0
Process 13190 terminated by signal 6
Retrying process 13190 with new process 13197 (attempt 1)
Process 13192 completed successfully. Exit code: 0
Process 13193 completed successfully. Exit code: 0
Process 13195 completed successfully. Exit code: 0
Process 13196 completed successfully. Exit code: 0
Child process 13197 crashed
Process 13197 terminated by signal 6
Retrying process 13197 with new process 13199 (attempt 2)
Child process 13199 crashed
Process 13199 terminated by signal 6
Retrying process 13199 with new process 13201 (attempt 3)
Process 13201 completed successfully. Exit code: 0
==============================================
Multi_process multiplication took 0.042948 seconds.
==============================================
Thread 140402078250752 crashed
Thread 140402205165312 completed successfully. Exit code: 0
Thread 140402196772608 completed successfully. Exit code: 0
Thread 140402078250752 terminated. Retrying...
Retrying thread 140402078250752 with new thread (attempt 1)
Thread 140402188379904 completed successfully. Exit code: 0
Thread 140402179987200 completed successfully. Exit code: 0
Thread 140402171594496 completed successfully. Exit code: 0
Thread 140402163201792 completed successfully. Exit code: 0
Thread 140402069858048 completed successfully. Exit code: 0
Thread 140402061465344 completed successfully. Exit code: 0
Thread 140402053072640 completed successfully. Exit code: 0
Thread 140402078250752 completed successfully. Exit code: 0
Final attempt to calculate row 2.
==============================================
Multi_threaded multiplication took 0.000956 seconds.
==============================================
** Single and multi_proc results are matched. **
==============================================
** Single and multi_thread results are matched. **
==============================================
** Multi_process and multi_thread results are matched. **
```

Test 8



```
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat2.txt 20
Child processes' crash rate: 20%
==============================================
matrix size: A- 10 * 20, B- 20 * 30
==============================================

==============================================
Serial multiplication took 0.000014 seconds.
==============================================

Child process 13740 crashed
Child process 13742 crashed
Child process 13744 crashed
Child process 13746 crashed
Process 13740 terminated by signal 6
Retrying process 13740 with new process 13755 (attempt 1)
Process 13741 completed successfully. Exit code: 0
Child process 13755 crashed
Process 13742 terminated by signal 6
Retrying process 13742 with new process 13757 (attempt 1)
Process 13743 completed successfully. Exit code: 0
Process 13744 terminated by signal 6
Child process 13757 crashed
Retrying process 13744 with new process 13758 (attempt 1)
Process 13745 completed successfully. Exit code: 0
Process 13746 terminated by signal 6
Retrying process 13746 with new process 13760 (attempt 1)
Process 13747 completed successfully. Exit code: 0
Process 13748 completed successfully. Exit code: 0
Process 13750 completed successfully. Exit code: 0
Child process 13760 crashed
Process 13755 terminated by signal 6
Retrying process 13755 with new process 13762 (attempt 2)
Child process 13762 crashed
Process 13757 terminated by signal 6
Retrying process 13757 with new process 13764 (attempt 2)
Process 13758 completed successfully. Exit code: 0
Process 13760 terminated by signal 6
Retrying process 13760 with new process 13765 (attempt 2)
Process 13762 terminated by signal 6
Retrying process 13762 with new process 13766 (attempt 3)
Process 13764 completed successfully. Exit code: 0
Process 13765 completed successfully. Exit code: 0
Process 13766 completed successfully. Exit code: 0
==============================================
Multi_process multiplication took 0.050926 seconds.
==============================================
Thread 140609906009856 crashed
Thread 140609889224448 crashed
Thread 140610011703040 completed successfully. Exit code: 0
Thread 140610003310336 completed successfully. Exit code: 0
Thread 140609914402560 completed successfully. Exit code: 0
Thread 140609994917632 completed successfully. Exit code: 0
Thread 140609906009856 terminated. Retrying...
Retrying thread 140609906009856 with new thread (attempt 1)
Thread 140609897617152 completed successfully. Exit code: 0
Thread 140609889224448 terminated. Retrying...
Retrying thread 140609889224448 with new thread (attempt 1)
Thread 140609880831744 completed successfully. Exit code: 0
Thread 140609872439040 completed successfully. Exit code: 0
Thread 140609864046336 completed successfully. Exit code: 0
Thread 140609906009856 completed successfully. Exit code: 0
Thread 140609889224448 completed successfully. Exit code: 0
Final attempt to calculate row 4.
Thread 140609889224448 crashed
Final attempt to calculate row 6.
Thread 140609889224448 crashed
==============================================
Multi_threaded multiplication took 0.000828 seconds.
==============================================
** Single and multi_proc results are NOT matched.**
==============================================
** Single and multi_thread results are NOT matched.**
==============================================
** Multi_process and multi_thread results are NOT matched.**
==============================================
```

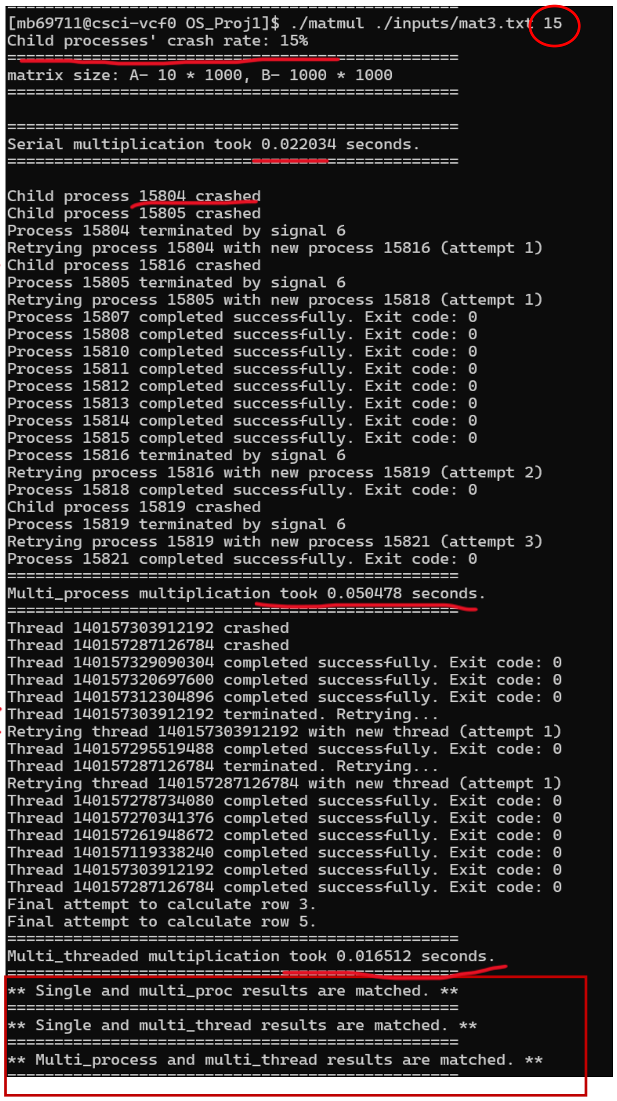
Test 7 and 8: In the mat2 input file I apply 20% crash rate several times to observe the performance. When a process or thread crashes, a new one is created, and the task is retried. The mismatch result may happen for many reasons. The system might retry a failed process or thread a limited number of times (3 attempts). If the process or thread consistently crashes after retries, the associated portion of the matrix may not be correctly computed, leading to mismatched results.

If a process or thread reaches the retry limit and still fails, that portion of the matrix will remain incomplete, which can cause the final result to differ from the correct computation. This explains why sometimes the results match, and other times they do not.

The crash rate (20%) introduces randomness into which processes or threads will fail during execution. Depending on which specific parts of the matrix computation are handled by the crashed process or thread, it can affect how much of the result matrix is impacted. In some cases, the portions that crash may be less critical or easier to recover, resulting in matched results. In other cases, more critical portions crash, leading to mismatches despite the same crash rate.

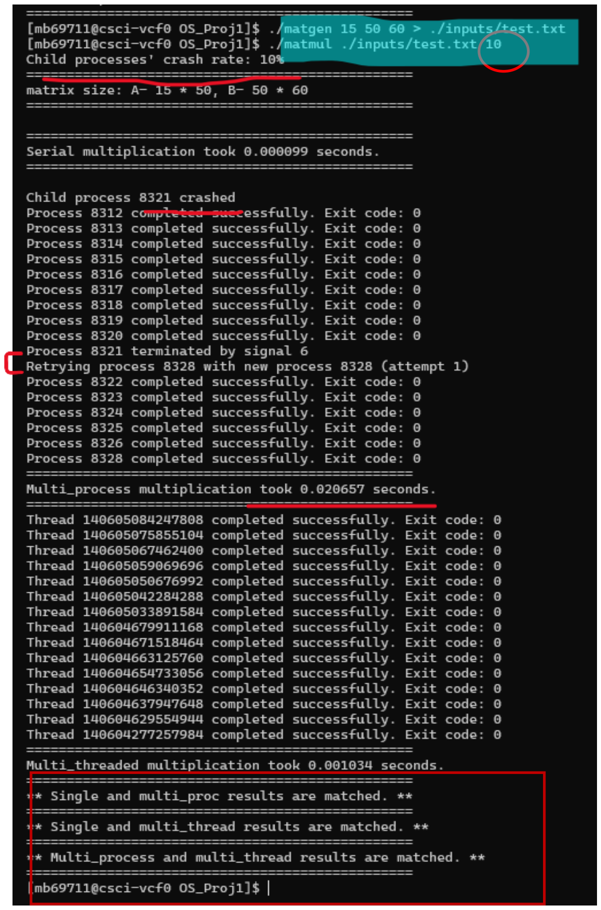
Name: Mayesha Bintha Mizan
ID-811281302

Test 9



```
================================================
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/mat3.txt 15
Child processes' crash rate: 15%
================================================
matrix size: A- 10 * 1000, B- 1000 * 1000
================================================

================================================
Serial multiplication took 0.022034 seconds.
================================================

Child process 15804 crashed
Child process 15805 crashed
Process 15804 terminated by signal 6
Retrying process 15804 with new process 15816 (attempt 1)
Child process 15816 crashed
Process 15805 terminated by signal 6
Retrying process 15805 with new process 15818 (attempt 1)
Process 15807 completed successfully. Exit code: 0
Process 15808 completed successfully. Exit code: 0
Process 15810 completed successfully. Exit code: 0
Process 15811 completed successfully. Exit code: 0
Process 15812 completed successfully. Exit code: 0
Process 15813 completed successfully. Exit code: 0
Process 15814 completed successfully. Exit code: 0
Process 15815 completed successfully. Exit code: 0
Process 15816 terminated by signal 6
Retrying process 15816 with new process 15819 (attempt 2)
Process 15818 completed successfully. Exit code: 0
Child process 15819 crashed
Process 15819 terminated by signal 6
Retrying process 15819 with new process 15821 (attempt 3)
Process 15821 completed successfully. Exit code: 0
================================================
Multi_process multiplication took 0.050478 seconds.
================================================
Thread 140157303912192 crashed
Thread 140157287126784 crashed
Thread 140157329090304 completed successfully. Exit code: 0
Thread 140157320697600 completed successfully. Exit code: 0
Thread 140157312304896 completed successfully. Exit code: 0
Thread 140157303912192 terminated. Retrying...
Retrying thread 140157303912192 with new thread (attempt 1)
Thread 140157295519488 completed successfully. Exit code: 0
Thread 140157287126784 terminated. Retrying...
Retrying thread 140157287126784 with new thread (attempt 1)
Thread 140157278734080 completed successfully. Exit code: 0
Thread 140157270341376 completed successfully. Exit code: 0
Thread 140157261948672 completed successfully. Exit code: 0
Thread 140157119338240 completed successfully. Exit code: 0
Thread 140157303912192 completed successfully. Exit code: 0
Thread 140157287126784 completed successfully. Exit code: 0
Final attempt to calculate row 3.
Final attempt to calculate row 5.
================================================
Multi_threaded multiplication took 0.016512 seconds.
================================================
** Single and multi_proc results are matched. **
================================================
** Single and multi_thread results are matched. **
================================================
** Multi_process and multi_thread results are matched. **
================================================
```

Test 10



```
================================================
[mb69711@csci-vcf0 OS_Proj1]$ ./matgen 15 50 60 > ./inputs/test.txt
[mb69711@csci-vcf0 OS_Proj1]$ ./matmul ./inputs/test.txt 10
Child processes' crash rate: 10%
================================================
matrix size: A- 15 * 50, B- 50 * 60
================================================

================================================
Serial multiplication took 0.000099 seconds.
================================================

Child process 8321 crashed
Process 8312 completed successfully. Exit code: 0
Process 8313 completed successfully. Exit code: 0
Process 8314 completed successfully. Exit code: 0
Process 8315 completed successfully. Exit code: 0
Process 8316 completed successfully. Exit code: 0
Process 8317 completed successfully. Exit code: 0
Process 8318 completed successfully. Exit code: 0
Process 8319 completed successfully. Exit code: 0
Process 8320 completed successfully. Exit code: 0
Process 8321 terminated by signal 6
Retrying process 8328 with new process 8328 (attempt 1)
Process 8322 completed successfully. Exit code: 0
Process 8323 completed successfully. Exit code: 0
Process 8324 completed successfully. Exit code: 0
Process 8325 completed successfully. Exit code: 0
Process 8326 completed successfully. Exit code: 0
Process 8328 completed successfully. Exit code: 0
================================================
Multi_process multiplication took 0.020657 seconds.
================================================
Thread 140605084247808 completed successfully. Exit code: 0
Thread 140605075855104 completed successfully. Exit code: 0
Thread 140605067462400 completed successfully. Exit code: 0
Thread 140605059069696 completed successfully. Exit code: 0
Thread 140605050676992 completed successfully. Exit code: 0
Thread 140605042284288 completed successfully. Exit code: 0
Thread 140605033891584 completed successfully. Exit code: 0
Thread 140604679911168 completed successfully. Exit code: 0
Thread 140604671518464 completed successfully. Exit code: 0
Thread 140604663125760 completed successfully. Exit code: 0
Thread 140604654733056 completed successfully. Exit code: 0
Thread 140604646340352 completed successfully. Exit code: 0
Thread 140604637947648 completed successfully. Exit code: 0
Thread 140604629554944 completed successfully. Exit code: 0
Thread 140604277257984 completed successfully. Exit code: 0
================================================
Multi_threaded multiplication took 0.001034 seconds.
================================================
** Single and multi_proc results are matched. **
================================================
** Single and multi_thread results are matched. **
================================================
** Multi_process and multi_thread results are matched. **
================================================
[mb69711@csci-vcf0 OS_Proj1]$ |
```

Test 9 & 10: In the mat3 input file I apply 15% crash and then generate a test matrix and apply 10% crash rate to observe the performance. Both the multi-process and multi-threaded approaches successfully handled crashes, demonstrating robust recovery mechanisms that ensure accuracy even when failures occur. In test 10, In the multi-process, one child process (Process 8321) crashed. This was immediately detected, and a new process (Process 8328) was created to replace the crashed one. In the multi-threaded, no threads crashed in this test.

The execution time and success of processes or threads could also be impacted by system load, available resources, and how the OS schedules tasks. Variability in execution timing and resource contention could cause different outcomes for identical crash rates.

In conclusion, the multi-process and multi-threaded models effectively handled matrix multiplication tasks, even under crash scenarios, with successful recovery mechanisms in place. Both approaches maintained accuracy, as all results matched the serial baseline, proving the system's reliability in handling parallel computations and crash recovery.