Operating Systems Laboratory (CS39002)
Assignment 3 - Task 1 [b]

This code is being run on a WSL (Windows Subsystem for Linux). The device has 8 GB RAM and 64-bit operating system (x64-based processor).

Here, we are multiplying two matrices of size r1 * c1 and r2 * c2. The resultant product matrix will be of size r1 * c2. We create r1 * c2 child processes, each for computing each element of the resultant matrix.

From the output given by the code, it is observed that we cannot compute the result beyond a limit as the fork() call fails, and the error is "Resource temporarily unavailable". This error is seen because of either of the following reasons:

- Limit on the maximum number of threads possible system-wide is *30771*. This can be checked by running the command:

```
cat /proc/sys/kernel/threads_max
```

- Limit on the maximum number of process identifiers (pids) that can be allocated by the system is *32768*. This can be checked by running the command:

```
cat /proc/sys/kernel/pid_max
```

- Limit on the maximum number of processes available to a single user is *15385*. This can be checked by running the command:

```
ulimit -u
```

The maximum number of processes that this system can run would be the minimum of the three values, i.e., `min(30771, 32768, 15385) = 15385`. Theoretically, the maximum size of matrix that can be multiplied by the system (in terms of r1 * c2) is *15385*. While executing the code, it is observed that the maximum number of child processes that could be created is 15228. So, we can say that the limit on the maximum number of processes available would have been reached, resulting in "Resource temporarily unavailable" error.

Even the above-mentioned value (*15385*) is a theoretical value. This is because the system runs many other processes, and all *15385* processes may not be available at a given instant. To show the same, the following command is executed:

```
ps -A --no-headers | wc -l
```

This gave *24* when the report is being made. So, the number of processes available would be `15385 - 24 = 15361`

But even then, the observed value while executing our code (*15228*) is lesser than *15361*. On running a simple C program that calls fork() as many times as it can, the number of times it is successful in doing so comes out to be *15215*. It does not go up to *15215* as the system doesn't let the program take up all the processes, and keeps some for itself.

Although we don't have the exact number, we get a good estimate that the maximum size of a matrix that can be multiplied by the system (in terms of r1 * c2) would be somewhere around *15200*, but will go at max *15385*.

The code runs successfully for the size *15228 (108 * 141)*, but it shows error for *15229 (97 * 157)*. This assures the correctness of the maximum size computed above.