There are several limitations to the number of processes we can fork. According to the man page of the fork() call the following errors may occur due to various limits:

```
EAGAIN A system-imposed limit on the number of threads was
encountered.  There are a number of limits that may trigger this
error:

*  the RLIMIT_NPROC soft resource limit (set via setrlimit(2)), which
limits the number of processes and threads for a real user ID, was
reached;

*  the kernel's system-wide limit on the number of processes and
threads, /proc/sys/kernel/threads-max, was reached (see proc(5));

*  the maximum number of PIDs, /proc/sys/kernel/pid_max, was reached
(see proc(5)); or

*  the PID limit (pids.max) imposed by the cgroup "process number"
(PIDs) controller was reached.
```

The most relevant to us are the pid_max and the RLIMIT_NPROC. They can be found using the following commands:

1. `cat /proc/sys/kernel/pid_max`
2. `cat /proc/self/limits`

Theoretically we cannot create more processes than these limits. So as an upper bound we must have ***r1*c2 <= min(pid_max, RLIMIT_NPROC)***.

On the system we tested the code on these limits were 32768 and 62780 respectively. So while it should be possible to create as many as 32768 processes we observed that the maximum processes running at any time could not go above 9000-9200. The fork() call failed whenever *r1*c2* went roughly above 8500 with approximately 500 processes already running in the system before the execution of our code.

This is primarily because of other limitations such as available memory which varies from system to system and hence the practical number of processes is actually lesser than the theoretical maximum possible.

References:

- https://stackoverflow.com/questions/29605502/maximum-number-of-children-processes-on-linux
- https://www.kernel.org/doc/man-pages