**Part II.1) : What does FORK() function do:**

In the code, fork() is a system call which creates a new process by duplicating the calling process. When fork() is called, the operating system copies the parent process's memory (so the code, data, stack, heap) and creates a new PCB for the child. The child process then also receives a unique PID, but at first it is nearly similar to the parent. The fork() returns two times. The first time is in the parent process and it returns the PID of the child. The second is in the child process and it returns 0. The difference allows programs to differentiate between parent and child and execute different code paths. For example, in the program here, the parent prints its own PID and the child's PID, the child prints that it will call exec() to transform itself into a whole other program. The fork() creates a separate process, so after fork, both parent and child will keep on executing the same code from the point fork was called, but they are now independent processes with different address spaces. Which is why the parent can keep running its loop while the child calls exec() to load process2. The output of the code also confirms this, because teh parent shows its PID and the child's PID, while the child says that it will exec into process2. Fork() is really important because it allows programs like shell to start new processes without affecting its own execution.

**Part II.2) : What does EXEC() function do:**

In the code, exec() is a system call that replaces the current process's memory with a new program while also containing the same PID. When the child process PID 495 calls exec("./process2", "process2", NULL), the operating system loads the process2 executable directly from the disk and changes the child's whole memory (code, data, stack, heap) with process2's content. What is different from fork() which creates a new process is that exec() transforms the current process. The child keeps PID 495 but now runs process2 code instead of process1. The output of the code also confirms this because the child says "will exec process2" and right after, "Process 2 starts with exec()" is with the same PID 495. This shows that the transformation happened. The fork() and exec() combination allows Process 1 to launch Process 2 as an independent program so not connected from the previous one. Without it affecting itself, this is important in how shells launch commands in UNIX systems.