1. What happens to your shell when you run commands that crash or prematurely terminate? Does your shell crash when a command crashes?
    a. When I run a command that isn't an actual command then my tinyShell doesn't crash, instead it just outputs `bash: command not found`
2. Determine system calls that are used in implementing the system() library function
    a. I noticed that `execve` is used to run my command. I chose to implement my version of system() by using execvp because it was easier to parse my commands into an array
3. Timing:
    a. Introduction: I measure the time in each implementing by looking at the time before the forking and then after the forking and subtracting them. I used a helper method called gettime() which was suggested to us in the tutorial.
    b. Fork
        i. I found that the timings really depended on which commands I ran.
            1. ~pwd: 2.970300e+04
            2. ~ls: 2.36250e+03
            3. ~date : 1.43200e+03
            4. ~echo hi: 7.25100e+03
            5. ~more Latim-Lipsum.txt: 1.901500e+03
    c. VFork
        i. To be honest, Vfork did not come out significantly faster than fork.
            1. ~pwd: 4.255000e+03
            2. ~ls: 3.466000e+03
            3. ~date: 1.466400e+04
            4. ~echo hi: 4.997000e+03
            5. ~more Latim-Lipsum.txt: 1.702300e+03
    d. Clone
        i. To be perfectly honest, I chose the flags in clone() by looking at the example given to us by the TA. But later I did some research about the flags to pass. Passing CLONE_FS makes sense because if it isn't set then the child process will work on a copy of the filesystem information. This means that some of the calls would not work (such as chroot, chdir, or unmask). CLONE_VFORK makes sense because execution will be suspended until the child releases its VM resources. This means that I wouldn't run out of memory in case I was passed a lot of commands in a single file.
        ii. Timings:
            1. ~pwd: 3.004923e+07
            2. ~ls: 2.828580e+6
            3. ~date: 6.042924e+07
            4. ~echo hi: 9.722323e+07
            5. ~more Latim-Lipsum.txt: 2.508856e+07

4. My parent waits for the child by waiting for the status to change → this is because the child's status will change from 0 when it dies.
5. Piping: I made my pipe work in a pretty simple way. The args specify whether the pipe is being read from or written to. Within my c file I have two methods, one for each action. Regardless of the action, my child closes either stdin or stdout. This avoids the problem of having to reset to stdout or stdin afterwards and avoids the cleanup.
6. Issues I had:
    a. I had some issues with exiting. I'm not sure why but it doesn't really work even though I tried to deal with it in my code. I set a condition that exits if the command is exit but my code doesn't exit
    b.