Title 8. Write a C/C++ program to avoid zombie process by forking twice.

- **If a child process terminates while its parent is calling a wait function, the child process vanishes and its termination status is passed to its parent via the wait call.**

- **If the child process finishes before the parent process calls wait, the child process becomes a zombie.**

- **When the parent process calls wait, the zombie child's termination status is extracted, the child process is deleted.**

However, sometimes we do not want the parent process to wait for its child process for a long time.

There is a way to achieve both "not create zombie process" and "not wait for the child process to its termination", and the way is to do a double fork.

How it works:

- Parent forks 1st child and waits for the child.
- The 1st child forks again to create second child.
- We call sleep in the second child to ensure that the first child terminates before printing the parent process ID.
- As soon as the first child dies 2nd will re-parented to init.
- Now parent of 2nd child will be init whose process id will be 1

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
    int pid;
    pid = fork();

    if (pid == 0)
    {
        // First child
        pid = fork();
        if (pid == 0)
        {
            // Second child
            sleep(1);
            printf("Second child: My parent PID is %d\n", getppid());
        }
    }
    else
    {
        // Parent process
        wait(NULL);
        sleep(2);
        system("ps -o pid,ppid,state,tty,command");
    }
    return 0;
}
```

SAMPLE OUTPUT

You try!!!!

THANK YOU