Que 2

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/wait.h>

int main() {

int call\_count = 0; // Counter for the number of calls

while (call\_count < 5) { // Limiting to 5 calls for demonstration

call\_count++;

printf("\nCall number: %d\n", call\_count);

if (call\_count % 2 == 0) {

pid\_t pid = fork(); // Only fork on even calls

if (pid < 0) {

// Fork failed

perror("Fork failed");

exit(1);

} else if (pid == 0) {

// Child process

printf("Child process created with PID: %d for call %d\n", getpid(), call\_count);

exit(0); // Child exits immediately

} else {

// Parent process

printf("Parent process with PID: %d created a child with PID: %d\n", getpid(), pid);

if (call\_count == 2) {

// Intentionally don't wait for the second child to create a zombie

printf("Intentionally leaving child process with PID %d as zombie.\n", pid);

} else {

// Wait for other child processes to prevent zombies

wait(NULL);

}

}

} else {

// For odd calls, no child process is created

printf("No child process created for this odd call.\n");

}

sleep(1); // Adding a sleep to observe the process states

}

// Final sleep to keep parent running for a while so zombie state can be observed

printf("\nSleeping for 10 seconds to observe the zombie process.\n");

sleep(10);

// Parent waits for all child processes to clear up zombies before exiting

while (wait(NULL) > 0);

printf("Parent process exiting. All child processes reaped.\n");

return 0;

}