

**NATIONAL TEXTILE**

**UNIVERSITY**

DEPARTMENT OF COMPUTER SCIENCE

**SUBMITTED BY:**

Farah Naz 23-NTU-CS-1152

**SECTION SE: 5th(A)**

**LAB MANUAL**

**SUBMITTED TO:**

Sir Nasir Mehmood

**SUBMISSION DATE:**

10-3-2025

**Task 1:**

**Code:**

 #include <stdio.h>

 #include <unistd.h>

 int main() {

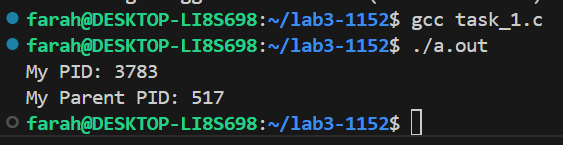
 printf("My PID: %d\n", getpid());

 printf("My Parent PID: %d\n", getppid());

 return 0;

 }

**Output:**



**Task 2:**

**Code:**

#include <stdio.h>

 #include <unistd.h>

 int main() {

 pid\_t pid = fork();

 if (pid == 0) {

 // This block runs in the child process

 printf("Child: PID=%d, Parent=%d\n", getpid(), getppid());

 } else {

 // This block runs in the parent process

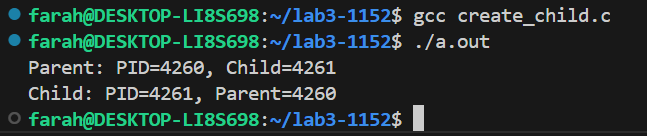
 printf("Parent: PID=%d, Child=%d\n", getpid(), pid);

}

 return 0;

 }

**Output:**



**Task 3:**

**Code:**

 #include <stdio.h>

 #include <unistd.h>

 int main() {

 pid\_t pid = fork();

 if (pid == 0) {

 execlp("ls", "ls", "-l", NULL);

 printf("This will not print if exec succeeds.\n");

 } else {

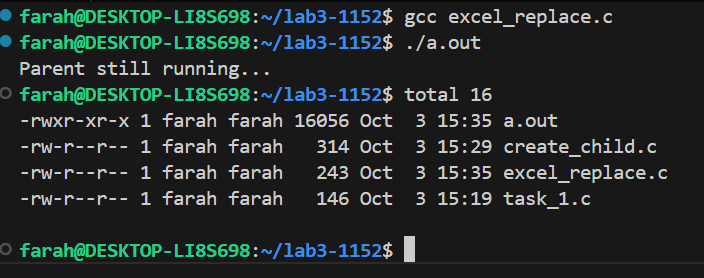
 printf("Parent still running...\n");

 }

 return 0;

}

**Output:**



**Task 4:**

**Code:**

#include <stdio.h>

 #include <unistd.h>

 #include <sys/wait.h>

 int main() {

 pid\_t pid = fork();

 if (pid == 0) {

 execlp("ls", "ls", "-l", NULL);

 printf("This will not print if exec succeeds.\n");

 } else {

 waitpid(pid, NULL, 0); // Wait for the child process to finish

 printf("Parent still running...\n");

 }

 return 0;

 }

**Output:**

