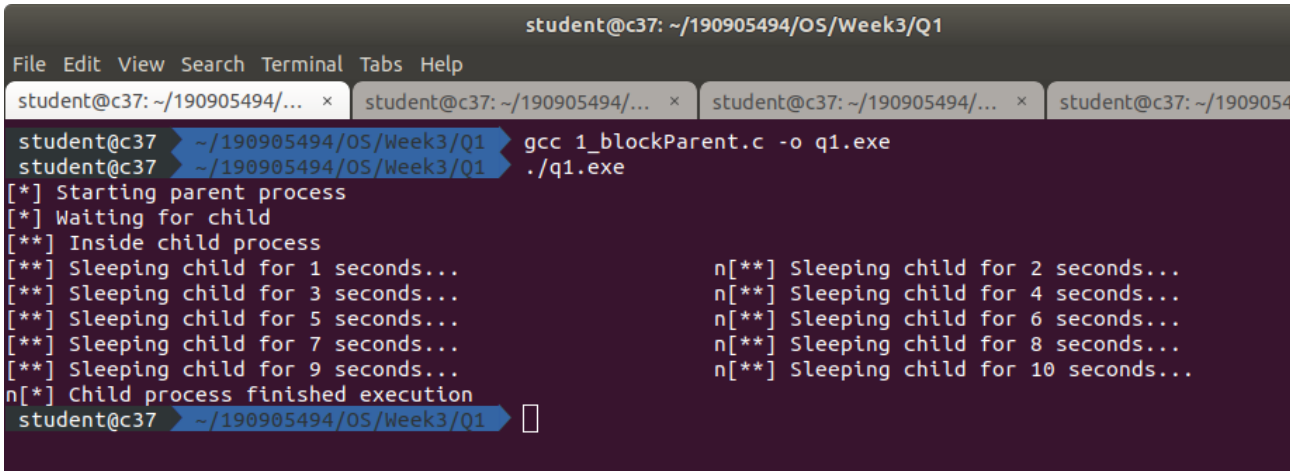
OS Lab #3

Question 1

```
/*
C Program to block a parent process, until
the child completes using a wait system call
*/

// imports
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main() {
   // creating child and parent process
   // storing process id in `r_val`
   printf("[*] Starting parent process\n");
   int r_val = fork();

   // switching between different processes
   switch(r_val) {

      // in case some error occurs in creation
      case -1: {
            perror("[*] fork\n");
            exit(EXIT_FAILURE);
         }
         break;

      // in case of child process
      case 0: {
            printf("[**] Inside child process\n");

            // Sleeping...
            for(int i = 0; i < 10; i++) {
               sleep(1);
               printf("[**] Sleeping child for %d seconds...\n", i + 1);
            }

            // exiting from the child program
            exit(0);
         }
         break;

      // in case of parent class
      default: {
            printf("[*] Waiting for child\n");
```

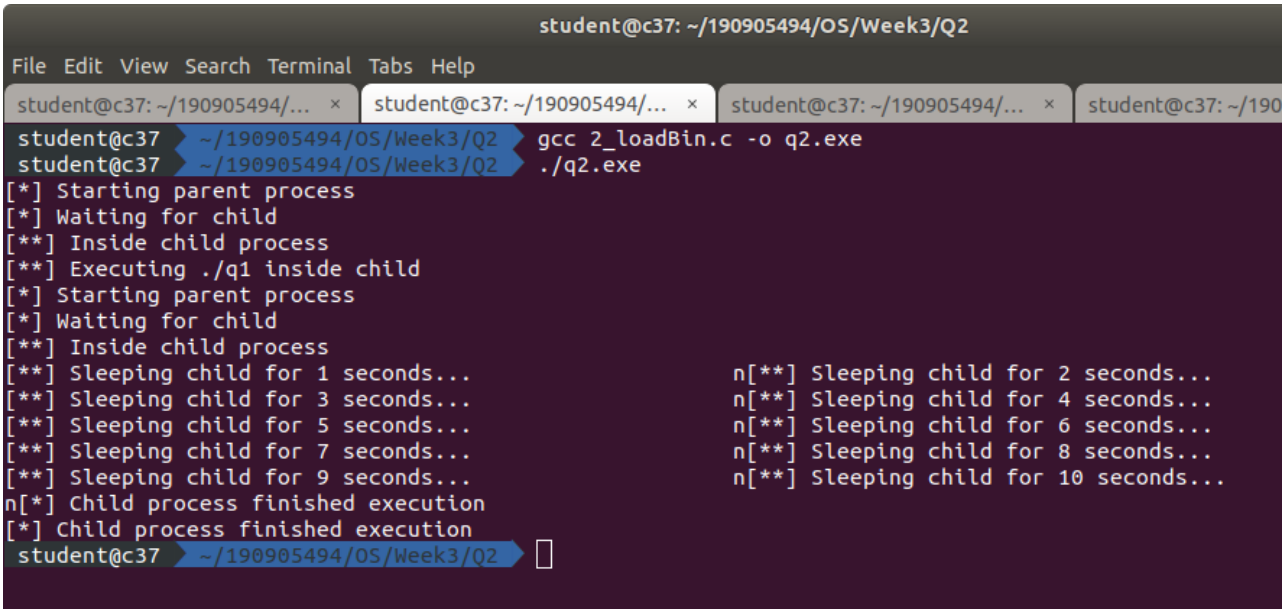
```
            // waiting for child process to finish executing
            wait(NULL);
            printf("[*] Child process finished execution\n");

            // exiting from the parent program
            exit(0);
        }
    }
}
```



Question 2

```
/*
Program to load binary executables of the previous program,
in a child process using `exec` system call
*/

// imports
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main() {
    // creating child and parent process
    // storing process id in `r_val`
    printf("[*] Starting parent process\n");
    int r_val = fork();

    // switching between different processes
    switch(r_val) {

        // in case some error occurs in creation
        case -1: {
            perror("[*] fork\n");
```

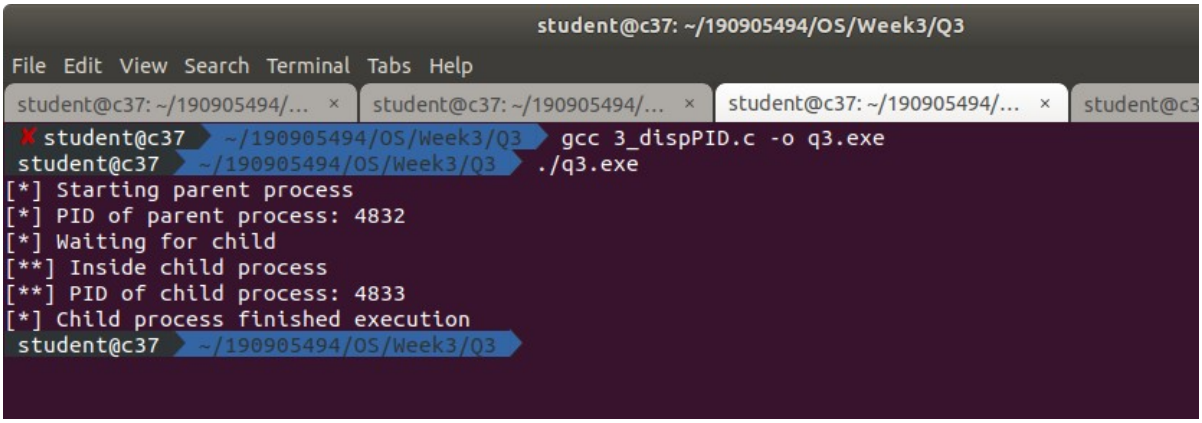
```
            exit(EXIT_FAILURE);
        }
        break;

    // in case of child process
    case 0: {
            printf("[**] Inside child process\n");
            printf("[**] Executing ./q1 inside child\n");

            // executing the output of the `1_blockParents.c` file
            execl("../Q1/q1.exe", "./q1.exe" , NULL);

            // exiting from the child program
            exit(0);
        }
        break;

    // in case of parent class
    default: {
            printf("[*] Waiting for child\n");

            // waiting for child process to finish executing
            wait(NULL);
            printf("[*] Child process finished execution\n");

            // exiting from the parent program
            exit(0);
        }
  }
}
```



Question 3

```
/*
Program to create a child process.
Display the process IDs of the process,
parent and child(s) in both the
parent and child process.
*/

// imports
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main() {
    // creating child and parent process
    // storing process id in `r_val`
    printf("[*] Starting parent process\n");
    int r_val = fork();

    // switching between different processes
    switch(r_val) {

        // in case some error occurs in creation
        case -1: {
            perror("[*] fork\n");
            exit(EXIT_FAILURE);
        }
        break;

        // in case of child process
        case 0: {
            printf("[**] Inside child process\n");

            // getting and printing PID
            pid_t curr_pid = getpid();
            printf("[**] PID of child process: %d\n", curr_pid);

            // exiting from the parent program
            exit(0);
        }
        break;

        // in case of parent class
        default: {
            // getting and printing PID
            pid_t curr_pid = getpid();
            printf("[*] PID of parent process: %d\n", curr_pid);
```

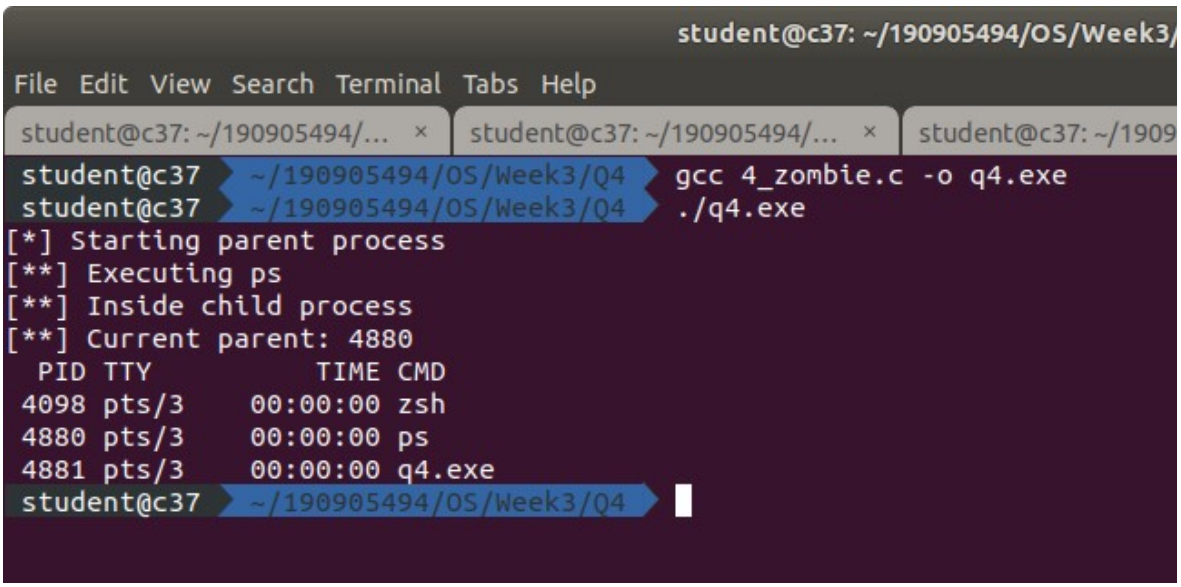
```
        // waiting for child process to finish executing
        printf("[*] Waiting for child\n");
        wait(NULL);
        printf("[*] Child process finished execution\n");

        // exiting from the parent program
        exit(0);
    }
  }
}
```



Question 4

```
/*
Create a zombie hild process,
and allow the init process to adopt it.
Run the process as a background process
and run the "ps" command
*/

// imports
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


int main() {

    // creating child and parent process
    // storing process id in `r_val`
    printf("[*] Starting parent process\n");
    int r_val = fork();

    // switching between different processes
    switch(r_val) {
```

```
        // in case some error occurs in creation
        case -1: {
            perror("[*] fork\n");
            exit(EXIT_FAILURE);
        }
        break;

        // in case of child process
        case 0: {
            printf("[**] Inside child process\n");

            // getting and printing PID
            pid_t curr_ppid = getppid();
            printf("[**] Current parent: %d\n", curr_ppid);

            // Sleeping...
            for(int i = 0; i < 10; i++) {
                sleep(1);
                printf("[**] Sleeping child for %d seconds...\n", i + 1);
            }

            // getting and printing PID
            curr_ppid = getppid();
            printf("[**] Current parent: %d\n", curr_ppid);

            // exiting from the child program
            exit(0);
        }
        break;

        // in case of parent class
        default: {
            // showing the current running process
            printf("[**] Executing ps\n");
            execl("/bin/ps", "ps", NULL);

            // exiting from the parent program
            exit(0);
        }
    }
}
```


student@c37: ~/190905494/OS/Week3/
File Edit View Search Terminal Tabs Help
student@c37: ~/190905494/... ×
student@c37: ~/190905494/... ×
student@c37: ~/1909
student@c37 ~/190905494/OS/Week3/Q4 gcc 4_zombie.c -o q4.exe
student@c37 ~/190905494/OS/Week3/Q4 ./q4.exe
[*] Starting parent process
[**] Executing ps
[**] Inside child process
[**] Current parent: 4880
PID TTY TIME CMD
4098 pts/3 00:00:00 zsh
4880 pts/3 00:00:00 ps
4881 pts/3 00:00:00 q4.exe
student@c37 ~/190905494/OS/Week3/Q4