Name: K Srirama Srikar
Roll No.: 142301013
Lab: 10

```
142301013 @ Lab 10 ($) gcc -o my_shell my_shell.c
142301013 @ Lab 10 ($) ./my_shell
Welcome to MY Shell!!
This is a coustom shell implemented in C!
Say hello and wait for magic to happen 😄
Type 'help' to get list of commands!!
my_shell> help
Available commands:
hello
add
subtract
cd
history
quit
Type help <command> to get more information about a specific command
my_shell> hello
Hello, welcome to my shell!
my_shell>
```

As we can see we get a list of commands when we type help and if we type help <command> we get the details of the command.

```
my_shell> add 10 20
Sum: 30
my_shell> subtract 30 40
Difference: -10
my_shell> help subtract
subtract <num1> <num2>: Subtracts the second integer from the first.
my_shell> sleep 60 &
Process running in background with PID: 6382
my_shell> ps
    PID TTY          TIME CMD
   5561 pts/1    00:00:00 bash
   6342 pts/1    00:00:00 my_shell
   6382 pts/1    00:00:00 sleep
   6383 pts/1    00:00:00 ps
my_shell> kill 6382
my_shell> ps
    PID TTY          TIME CMD
   5561 pts/1    00:00:00 bash
   6342 pts/1    00:00:00 my_shell
   6388 pts/1    00:00:00 ps
my_shell>
```

As we can see we are able to create processes in the background and when we send a signal to the process in the background we use the signal_handler function with the SIGCHILD parameter to make sure that there wont be any zombie processes running

```
my_shell> quit
Thank you for using the shell...
Bye 😉
142301013 @ Lab 10 ($)
```

Belo are the details of the implemented functions

1. **`print_help(char **args)`**
    - **Purpose:** Provides a help menu listing available commands or detailed help for a specific command.
    - **Usage:** `help` for general help, `help <command>` for specific help.

2. **`execute_external(char **args, int is_background)`**
    - **Purpose:** Executes external Linux commands using the `execvp()` function.
    - **Working:**
        - A child process is created using `fork()`.
        - If it's a background process (ends with `&`), we check using the `is_background` variable to see if it is to be executed in background or not, the parent does not wait for it to complete.

3. **`signal_handler(int sig)`**
    - **Purpose:** Handles `Ctrl+C` (SIGINT) and prevents zombie processes (SIGCHLD).
    - **Working:**
        - On SIGINT, we print a message and continue running the shell and donot terminate it.
        - When we pass SIGCHLD as the parameter, the parent process reaps terminated child processes to avoid zombies.