Problem Statement: Custom Shell Implementation
In this assignment, you will create a custom shell in C that can execute various commands and handle different functionalities such as command execution, command redirection, parallel command execution, sequential command execution, and signal handling. The shell will mimic a subset of the functionalities provided by standard Unix shells. Your implementation will include parsing user input, forking processes to execute commands, handling built-in commands like cd, and implementing custom features for parallel and sequential command execution.

Functional Requirements
Command Execution:
The shell should be able to execute single commands entered by the user.
Use fork() to create a child process and execvp() to execute the command in the child process.

Change Directory (cd) Command:
Implement the built-in cd command to change the current working directory.
Handle errors gracefully if the directory does not exist.

Exit Command:
Implement the exit command to terminate the shell.

Command Redirection:
Support output redirection using the > symbol.
Redirect the output of a single command to a specified file.

Parallel Command Execution:
Support the execution of multiple commands in parallel using the && symbol.
Commands separated by && should be executed simultaneously.

Sequential Command Execution:
Support the execution of multiple commands sequentially using the ## symbol.
Commands separated by ## should be executed one after another.

Signal Handling:
Implement signal handlers for SIGINT (Ctrl+C) and SIGTSTP (Ctrl+Z).
Ensure the shell displays the prompt correctly after handling these signals.
Implementation Details

Parsing Input:
The parseInput function takes a line of input and splits it into tokens (words) based on spaces, tabs, and newlines.
It returns an array of strings (tokens) which represent the command and its arguments.

Signal Handling:
Implement SIGINThandler and SIGSTPhandler functions to handle SIGINT and SIGTSTP signals.These handlers should print the current working directory and the shell prompt when a signal is received.

Command Execution:
The executeCommand function forks a new process to execute a single command using execvp.

Command Redirection:
The executeCommandRedirection function handles commands with output redirection (>). It redirects the output of the command to a specified file.

Parallel and Sequential Execution:
The executeParallelCommands function handles commands separated by &&, executing them in parallel.
The executeSequentialCommands function handles commands separated by ##, executing them sequentially.

Main Loop:
The main loop of the shell repeatedly prompts the user for input, parses the input, and determines which function to call based on the parsed tokens.
It handles built-in commands like cd and exit, and checks for special symbols (>, &&, ##) to decide the appropriate execution path.