Assignment 1
Suggested Approach for a Single Pipe

- The first process (parent shell) should fork one child process.
- The parent shell should wait for this child process to complete.
- The child process is the parent of all other processes where each of these processes executes a command.
- Let’s look at an example for `ps -l | sort`
Example

```
sHELL (0) -----> shell (1) -----> shell (2) -----> ps -el
           |               | exec
           |               | +-----> pipeline 1
           |               |
           |               |        +-----> sort
           |               | exec

fork
fork
fork
```
Example

- The parent shell (0) forks one child process (1). It waits for that child to terminate.
- The child process (shell 1) of the first step forks off two other processes: shell 2 and shell 3.
  - Each process redirects STDIN and STDOUT to the appropriate pipe and then calls `exec()` to execute the proper command.
Another Approach

- The parent shell (0) forks one child process (1). It waits for that child to terminate.

- The child process (shell 1) of the first step forks off one other process: shell 2
  - Both processes redirect STDIN and STDOUT to the appropriate pipe and then calls `exec()` to execute the proper command
  - The shell 1 process executes the second command
I/O Redirection

- We have discussed this already;
- Basically you will redirect standard input and output based using dup2
File Descriptor Considerations

- The process that is the child of the shell (shell 1) is responsible for creating all the needed pipes before it forks off any of its children.
  - This means that each of the children of shell 1 has a set of file descriptors for all pipes in the total pipeline.
File Descriptor Considerations

- Each forked process must specify exactly which pipe ends become its `stdin` and `stdout`.
  - For example, for `ls` the output should be associated with the pipeline 1.
  - Use `dup2(pipefd, stdin)` or `dup2(pipefd, stdout)`.

- Each forked process must close all file descriptors that comprise its pipes so that the pipes do not hang.
Multiple pipes

- The first process (parent shell) should fork one child process.
- The parent shell should wait for this child process to complete.
- The child process is the parent of all other processes where each of these processes executes a command.
- Let’s look at an example for
  
  ```
  ls -l | sort | more
  ```
Example
Example

- The parent shell (0) forks one child process (1). It waits for that child to terminate.
  - This child process (shell 1) executes the last command i.e., the `more` command.
- The child process (shell 1) of the first step forks off two other processes: shell 2 and shell 3.
  - Each new child process redirects STDIN and STDOUT to the appropriate pipe and then calls `exec()` to execute the proper command.
Example

- shell 2’s exec call loads the ls binary
- shell 3’s exec call loads the sort binary
- shell 1’s exec call loads the more binary
  - When it terminates it sends a notification to the parent shell
- The parent shell must wait on the last command to finish before continuing