1.
To ensure that commands are executed regardless of whether each previous command succeeds, we would parse “;” as symbols. Fork and make a child process for the command before the “;” symbol. In the meantime, parent process goes ahead with the other commands after the “;” symbol. The processes do not wait on each other.

```
Parse ‘;’
Case ‘;’:
        if(fork() == 0){
                execute command
        }else{
        // parent process
                execute command
                // no wait()
        }
```

To warrant that a command executes only if the previous command succeeds, we would parse “&&” as symbols. Fork and make a child process for the command before “&&” symbol. After the child process returns a success code, the parent process would proceed with the other commands which are after the “&&” symbol. If the child process fails, it exits with error.

```
Parse ‘&&’
Case ‘&&’:
        if(fork() == 0){
        // child process
                execute command
        }else{
        // parent process
                wait for child process
                check for execution return code
                if successfully executed, proceed with the rest
                if failed, exit with error
        }
```

2.
Parse ‘(’ and ‘)’ as symbols.
When there is a ‘(’, create a sub shell to execute the commands within ‘(’ and ‘)’.
After executing the commands in the sub shell, return the output of the sub shell to be used in the parent’s shell.

```
Parse ‘(’ and ‘)’
Case ‘(’:
        create a sub shell
        create a pipe and change output to the file descriptor of the pipe
        execute commands
```

close and exit sub shell
change input to the file descriptor of the pipe
execute the rest of the commands

3.
Parse ‘&’ and ‘wait’ as symbols.
When there is a ‘&’ in the command, create a child process to execute the prior command, and the parent process would execute the later command in parallel.
When there is a “wait”, wait for the process to finish before proceeding to anything else.

Parse ‘&’ and ‘wait’
Case ‘&’:

```
if(fork1() == 0){
        // child process
        execute commands
}else{
        execute commands
        // no wait()
}
```

Case ‘wait’:
wait for commands