1. The shell supports most of the features of a normal shell. While it might be slightly different from the original shell, mainly:
2. There will be an output of table of jobs displayed whenever a process is completed.
3. It is only capable of handling two signals:

CTRL + Z: stops all running processes

CTRL +Ｃ: kills all processes

1. CTRL+C kills all processes rather than only killing the foreground process.’
2. Somehow the wc commands does not work, but other commands should be working properly.
3. Below is a list of commands that are guaranteed to work for testing out the features of my shell.
4. Internal commands:

cd

**cd** //By default, cd without any inputs will change the current working directory to home.

1. Jobs

**jobs** //This command will list a table of all jobs either running or stopped

1. fg

**sleep 10 &** //Start a job that runs in the background

**jobs** //Display the table of jobs

**fg 1** //Bring the job with job id 1 to foreground

1. bg

**sleep 10** **&** //Start a job in the foreground

**CTRL + Z** //Stop all running processes

**jobs** //Display the table of jobs

**bg 1** //Bring the job with job id 1 to background

1. pipes

**ps ax | grep bash** // List all process running on system and only show the ones that contain the word bash

List of references which I received help from (However, none of the codes from these websites are used in this assignment):

http://stackoverflow.com/questions/8827939/handling-arguments-array-of-execvp

https://gist.github.com/leehambley/5589953

http://stackoverflow.com/questions/6574370/job-control-in-linux-with-c

http://simplestcodings.blogspot.ca/2010/10/story-of-zombie-process.html