CSC 207 Assignment 2B
Due Date: November 20th 2015 @ 11:59pm

Introduction:

In Assignment2B, you will continue to work on the ***mock file system*** and ***JShell***. We have modified the commands a bit, hence read this document very carefully. Note that at end of Assignment2B, all commands and functionality that were mentioned in Assignment2A and now mentioned in Assignment2B must be completed. If you had any left over work from Assignment2A, your team must ensure that it is completed in Assignment2B. If you create any new Exception classes, make sure to update your CRC Cards accordingly with CRC Cards. You MUST also use Generics.

You will create a new *productBacklog* as *productBacklog2.txt* inside your *productBacklog* folder. As in Assignment2A, you will continue to follow the scrum software development process in 2B as well. At the end of Assignment2B, each one of you will be asked to evaluate your team members. More details on this later.

You may want to first read the **Assignment2B QandA** on blackboard before getting started with the assignment.

Commands:

In all of the following commands, there may be any amount of whitespace (1 or more spaces and tabs) between the parts of a command.

For example (all the **three** variations are correct and valid):

```
/#: mv someFile1          someFile2
/#: mv          someFile1                  someFile2
/#: mv someFile1 someFile2
```

Furthermore, your program must not crash. Square brackets indicate an optional argument.

Clarification: ... indicates a list.

Clarification: every *PATH*, *FILE*, and *DIR* may either be *relative path* or a *full path (absolute path).*

Addition: redirection, either with > *OUTFILE* or >> *OUTFILE*, can be applied to every command except for *exit*. This captures the output of the command and redirects it to OUTFILE and no output is shown to the user. MAKE SURE THAT YOU NOT REDIRECT ANY ERRORS. ALL ERROS MUST STILL BE DISPLAYED ON THE CONSOLE. YOU ONLY REDIRECT STDOUTPUT AND NOT STDERROR. IF YOU ARE

NOT FAMILIAR WITH STDOUTPUT AND STDERROR, PLEASE REFER TO YOUR LABS. If a command does not give any output (such as the *cp* command) then you must not create *OUTFILE*.

exit

Quit the program

mkdir *DIR …*

Create directories, each of which may be relative to the current directory or may be a full path.

cd *DIR*

Change directory to DIR, which may be relative to the current directory or may be a full path. As with Unix, **..** means a parent directory and a **.** means the current directory. The directory must be /, the forward slash. The foot of the file system is a single slash: /.

ls *[-R] [PATH …]*

Addition: if –R is present, recursively list all subdirectories.

If no paths are given, print the contents (file or directory) of the current directory, with a new line following each of the content (file or directory).

Otherwise, for each path p, the order listed:

- If p specifies a file, print p
- If p specifies a directory, print p, a colon, then the contents of that directory, then an extra new line.
- If p does not exist, print a suitable message.

pwd

Print the current working directory (including the whole path).

mv ***OLDPATH NEWPATH***

Addition: This is a new command in Assignment2B

Move item OLDPATH to NEWPATH. Both OLDPATH and NEWPATH may be relative to the current directory or may be full paths. If NEWPATH is a directory, move the item into the directory.

cp ***OLDPATH NEWPATH***

Addition: This is a new command in Assignment2B

Like mv, but don't remove OLDPATH. If OLDPATH is a directory, recursively copy the contents.

cat ***FILE …***

Addition: if there are more than one file, you must display all their contents on the console. (assuming all are valid path). For any file that contains invalid path, just display an appropriate error for that path only. All other valid path must still be shown on the console.

Display the contents of FILE and other files on the console in the shell.

get ***URL***

Addition: This is a new command in Assignment2B

URL is a web address. Retrieve the file at that URL and add it to the current working directory.

Example1:

get http://www.cs.cmu.edu/~spok/grimmtmp/073.txt

Will get the contents of the file i.e. 073.txt and create a file called 073.txt with the contents in the current working directory.

Example2:

get http://www.ub.edu/gilcub/SIMPLE/simple.html

Will get the contents of the file i.e. simple.html (raw html) and create a file called simple.html with the contents in the current working directory.

echo ***String***

Print String

man *CMD*

Print documentation for CMD.

pushd *DIR*

Saves the current working directory by pushing onto *directory stack* and then changes the new current working directory to DIR. The push must be consistent as per the LIFO behavior of a stack. The pushd command saves the old current working directory in *directory stack* so that it can be returned to at any time (via the popd command). The size of the *directory stack* is dynamic and dependent on the pushd and the popd commands.

popd

Remove the top entry from the *directory stack,* and cd into it. The removal must be consistent as per the LIFO behavior of a stack. The popd command removes the top most directory from the *directory stack* and makes it the current working directory. If there is no directory onto the stack, then give appropriate error message.

history [number]

This command will print out recent commands, one command per line. i.e.

```
1  cd ..
2  mkdir textFolder
3  echo "Hello World"
4  fsjhdfks
5  history
```

The above output from history has two columns. The first column is numbered such that the line with the highest number is the most recent command. The most recent command is history. The second column contains the actual command. Note: Your output should also contain as output any syntactical errors typed by the user as seen on line 4.

We can truncate the output by specifying a number (>=0) after the command. For instance, if we want to only see the last 3 commands typed, we can type the following on the command line:

```
history 3
```

And the output will be as follows:

```
4  fsjhdfks
5  history
```

```
6  history 3
```

!number

Addition: This is a new command in Assignment2B

This command will recall any of previous history by its number(>=1) preceded by an exclamation point (!). For instance, if your history looks like above, you could type the following on the command line of your JShell i.e.

```
!3
```

This will immediately recall and execute the command associated with the history number 3. The above command of !3 will indirectly execute `echo "Hello World"`.

grep *[-R] REGEX PATH …*

Addition: This is a new command in Assignment2B

If –R is not supplied, print any lines containing REGEX in PATH, which **must** be a file. If –R is supplied, and PATH is a directory, recursively traverse the directory and, for all lines in all files that contain REGEX, print the path to the file (including the filename), then a colon, then the line that contained REGEX.

In order to do well for this Assignment, you MUST strictly adhere to the SCRUM software development process just as Assignment2A. Follow the instructions carefully that are mentioned in the grading scheme (same grading scheme applies for Assignment2B and Assignment2A).