Simple Shell Tests

Stage 1

Test: Lack of input i.e., just hitting enter ☐
Result: prints prompt again

Test: Enter input > 512 characters ☐
Result: No seg fault, error message printed

Test: Enter input with more than 50 tokens ☐
Result: No seg fault, error message printed

Test: Enter “hello -pp -m” ☐
Result: Tokens should be “hello”, “-pp” “-m”

Test: Enter exit ☐
Result: Shell should exit

Test: Enter <ctrl-d> ☐
Result: Shell should exit

Test: Enter exit as first input ☐
Result: Shell should exit, no seg
fault or extra outputs

Test: Enter <ctrl-d> as first input ☐
Result: Shell should exit, no seg
fault or extra outputs

Test: Enter ‘ls-lF;.&..>.fksdk’ ☐
Result: Error for fksdk and listings of . and ..
twice plus listing of / all formatted with
entry per line and file permissions

Test: Characters used in tokenization ☐
Result: Should be (space \t \n ; & > < |)

Stage 2

Test: Enter ps aux ☐
Result: Should list all processes and their status and resource usage

Test: Enter clear ☐
Result: Should clear the simple shell screen

Test: Enter ls ☐

Result: Should list files and directories in bare format

Test: Enter ls -l ☐
Result: Should show file or directory, size, modified
Date and time, file or folder name and owner of file
And its permission

Test: Enter lksm ☐
Result: Should display error “Command <name> not known”

Test: Run shell within an existing shell then type <ctrl-d> ☐
Result: Should exit one shell but not the other

Test: Try to run a program in shell that doesn’t exist ☐
Result: Appropriate error message no crashes

Stage 3

Test: Enter pwd ☐
Result: Should display the users home directory

Test: Enter getpath ☐
Result: Should display current path

Test: Enter setpath /bin ☐
Result: Should set current path to /bin

Test: Check ls works in this path ☐
Result: Should work fine

Test: Check clear works in this path ☐
Result: Should output an error/or not work

Test: Exit path ☐
Result: Should print getenv return value for path

Test: Restore path on exit command ☐
Result: Should restore path

Test: Restore path on <ctrl-d> command ☐
Result: Should restore path

Test: Enter getpath with parameters ☐
Result: Should print appropriate error message

Test: Enter setpath with no parameters ☐

Result: Should print appropriate error message

Test: Enter setpath with 1 parameter ☐
Result: Should print appropriate error message

Test: Do error messages say what the problem is and how to fix it? ☐

Test: Print what HOME is when retireved and print ☐
current directory after changing to home
Result: Both printed directories should be the same

Test: Change HOME environment by using ‘set’ command ☐
Result: Check if this works as expected

Stage 4

Test: Enter cd <directory> ☐
Result: Should change directory to one entered

Test: Enter cd /bin ☐
Result: Should change directory to bin

Test: From /bin enter ../.bin/./.. ☐
Result: Should change directory to /

Test: Check the above with pwd and ls ☐
Result: Check if this works as expected

Test: Enter cd to a non-existant directory ☐
Result: Print “no such file or directory”

Test: Enter cd to an existing file ☐
Result: Print “not a directory”

Test: Does error messages contain the name of the file ☐
that caused the error.

Test: Enter cd with two or more parameters ☐
Result: should print error message

Test: Does error messages say what the problem is and how to fix it ☐

Stage 5

Test: Print history

Result:
- Print every command + its parameters ☐
- List all commands with a number in front of them ☐
- List commands in ascending order ☐
- Numbers list start from 1 ☐
- Last command at bottom of list with biggest no ☐
- Store history as the last command ☐
- No empty history positions. ☐
- Above apply when history is full ☐

Test: History invocations with positive numbers (!<no>)

Result:
- Invoke command and its parameters ☐
- Invocations work for internal commands ☐
- Invocations work for external commands ☐
- Invoking the same command multiple times ☐
- Invocations leave history unchanged ☐

Test: History invocations with negative numbers (!-<no>)

Result:
- Invoke command and its parameters ☐
- Invocations work for internal commands ☐
- Invoations work for external commands ☐
- Invoking the same command multiple times ☐
- Correct command invoked i.e., -1 last command ☐
- Invocations leave history unchanged ☐

Test: History error handling

Result:
- History invocation with first command clear history ☐
- Number given > commands in hisory but < maximum given ☐
- Negative number out of range – histroy invocation ☐
- History + parameters should print error ☐
- ! followed with something other than number ☐
- !! !<no> !-<no> followed by something else should print error message ☐
- Error messages clear of what went wrong ☐

Stage 6

Test: Open file containing history ☐
Result: Should show commands in right order

Test: Open file containing histroy when history is full ☐
Result: Same as above

Test: Is the file called .hist_list ☐

Test: Is the history file located in users home ☐

Test: Open history when history is first and last command ☐
Result: Should show correct distory listing

Test: Invoke command from history as the first command ☐
Result: Should work as expected

Test: Start shell from different directory ☐
Result: Shouldn’t crash

Test: Deleting the file history ☐
Result: Shouldn’t crash

Test: Empty histroy file ☐
Result: Shouldn’t crash

Test: Histroy file contents random (include numbers in file) ☐
Resut: Shouldn’t crash

Stage 7

Test: Alias command when list is empty (no aliases set) ☐
Result: Appropriate error message

Test: Set alias ☐
Result: Check if its been set correctely including command parameters

Test: Executing aliases
Result:
- Execution includes command parameters ☐
- Execute alias multiple times ☐
- Execute with additional parameters e.g. ☐
  alias la to execute ls -la then call la /
  which should execute ls -la /
- Use internal commands as aliases e.g. ☐
  cd to execute ps
- Alias internal commands e.g. ☐
  alias bin to execute cd /bin
- Are aliases listed in history (aliases not command) ☐
- Invoke alias from history ☐
- Parameters on invoking aliases e.g. alias ☐
  la to execute ls -la then call la / followed by history
  and then call !-2 that should execute la -la /

Test: Define an existing alias ☐
Result: Error message or a message for overrinding

Test: Overriding aliases ☐
Result: alias p to execute ps aux, then alias p to
Execute ps, run p that should execute just ps

Test: Unalias when list is not full ☐
Result: Removes the alias

Test: Unalias when list is full ☐
Result: Creates an opening (i.e. you can introduce new alias)

Test: Alias with 1 parameter ☐
Result: Should give an error message

Test: Attempt to alias when list is full ☐
Result: Give an error message

Test: Using aliasing an existing alias overrides this alias ☐
Result: Overriding should work when list is full i.e.
No error just overriding message

Test: Attempting to unalias when alias list is empty ☐
Result: Give an error message

Test: Attempting to unalias a non-existing alias ☐
Result: Give an error message

Test: Enter the maximum number of aliases and enter another one
Result:

Test: Are error messages clear on what the problem is ☐

Stage 8

Test: Opening file containing aliases ☐
Result Should show alias and aliases commands in the correct order

Test: Opening file containing aliases when list is full ☐
Result: Same as above

Test: Is the file called .aliases ☐

Test: Is the file located in the user’s home ☐

Test: Open file when alias is the first command ☐
Result: See correct aliases listed

Test: Invoke an alias from saved aliases as the first command ☐
Result: Should work as expected

Test: Invoke a command from saved history that refers to an alias ☐
Result: Aliased command should execute

Test: Invoke a command from saved history that refers to an alias ☐
(with parameters)
Result: Aliased command should execute with parameters

Test: Start shell from different directory ☐
Result: Shouldn't crash

Test: Start shell after deleting the aliases file ☐
Result: Shouldn't crash

Test: Change to / then exit ☐
Result: Shouldn't crash

Test: Aliases file is empty ☐
Result: Shouldn't crash

Test: Aliases file contains random text (i.e rubbish) ☐
Result: Shouldn't crash

Stage 9

Test: Set alias chains so that the first alias will execute the last in chain ☐

Test: Alias a history invocation correctly e.g. alais five !5 ☐

Test: Create alias history invocation ! -1 ☐
Result: This should always execute last history command
even as history changes

Test: Define alias a b then b a ☐
Result: Executing this should give an error message

Test: Error messages should be clear of what problem is ☐