2. Evaluation sheet

TIP
for a moment! Check out 42wiki/minishell manual testing .

2.1. Mandatory Part Mandatory Part

2.1.1. Simple Command & Global Simple Command & Global Variable

- Run simple commands with absolute paths and no options. (For example: **/bin/ls**)
- How many global variables did you use? why? Explain logically why this is necessary with a concrete example.

2.1.2. Arguments parameter

- Run simple commands with absolute paths and no options. (For example: **/bin/ls**)
  You must use parameters other than " and '.
- Repeat several times with different parameters!

2.1.3. echo

- Execute both when there are no parameters and when they exist.
- Repeat multiple times with different inputs!.

2.1.4. exit

- Execute both when there are no parameters and when they exist.
- Repeat multiple times with different inputs!.
- Of course, don't forget to run it again.

2.1.5. Return value of a process ($?)

- Try running simple commands like **/bin/ls** with parameters (no ", ').
  After running **echo $?** Try running
- Check the output value. See if it works the same in bash.

- Use different commands and parameters multiple times, and even use the wrong command.
  For example: **/bin/ls filethatdoesnotexist**
Footnotes in kkim: $? It seems that the number generated during execution can be known by studying the error code.

2.1.6. Semicolons Semicolon (;)

- Try using multiple simple commands (using absolute path commands) without parameters, separated by ;.
*Example of kkim: **/bin/ls;/bin/pwd***
Note from kkim: When separating instructions with ;, there is some debate about whether or not they run in parallel.
***cat ; Try running pwd** and think about it. Even in bash!*
- Test several times. Another command, as a parameter! Don't forget to remove the spaces around the semicolon.

2.1.7. Signals

- Try running **ctrl C** in an **empty prompt .**
- Try running **ctrl ** in an **empty prompt .**
- Try running **ctrl D** in an **empty prompt .**
- Try running **ctrl C** at **the prompt where you have written something .**
- Try running **ctrl ** at **the prompt with some writing on** it.
- Try running **ctrl D** at **the prompt with something written on** it.
- Try running **ctrl C** in **a blocked prompt by running cat, grep (without parameters)** .
- Try running **ctrl ** in **a blocked prompt by running cat, grep (without parameters)** .
- Try running **ctrl D** in **a blocked prompt by running cat, grep (without parameters)** .
- Repeat several times with different commands!

2.1.8. Double Quotes

- This time try running a simple absolute path command with parameters.
  But now, with double quotation marks
- Think about the wrong use of an empty parameter, \.
- Do not use multi-line strings.

2.1.9. env

- Check whether the actual environment variable is output.

2.1.10. export

- Modify or add environment variables.
- Also check the env!.

Note from kkim: Although not written in the evaluation table, if no parameters are entered, a list of environment variables is printed with the message ***declare -x .*** *But.. the sorting method is different from env..*

2.1.11. unset

- *Is this an error..?* Modify or add environment variables.
- Try deleting some with unset.
- Check the result with env! (You can also check it by export haha)

2.1.12. Environment Variables

- Try printing the $environment variable through echo.
- Make sure it works even when surrounded by "" like in bash.

Note from kkim: it doesn't work inside ", but works with "". After $, it checks until an empty character comes, and the first letter is an alphabet, and the last letter is alphabet + numbers. Then, if there is a character before $...?

2.1.13. CD

- Try moving the working directory using the cd command.
  Also check that /bin/ls is in the proper directory.
- Check several times by putting a working or inoperative (invalid) directory as well.
- Also check . and ..

2.1.14. pwd

- Use the pwd command.
- Try using it several times, in another directory.

2.1.15. Relative Path

- Try running the command with a relative path.
- Run multiple times with complex paths in different directories.

2.1.16. Simple Quotes single quotes (')

- Try putting single quotation marks around the parameter.
- Also test empty parameters.
- Also test environment variables, spaces, and semicolons. (should not work)

2.1.17. Redirection

- Try using commands with redirects. <, >.
- Test with different commands and parameters several times. Sometimes > and >> too!
- Check if the same redirect is repeated multiple times or not.

2.1.18. Pipes

- **cat file | grep bla |** Try putting in a pipe like **more and so on.**
- Test with different commands and parameters several times
- Invalid command **ls filethatdoesnotexist | grep bla |** Also test invalid commands like **more !**

2.1.19. Go Crazy and history Let's go crazy! And history...

- Check if you can guide the history through **up** and **down .** many times!
- Try entering commands that do not work (ex: **dsbksdgbksdghsd**), and if the shell does not crash, print an error.
- Test really really really really long commands and parameters.
- ...I hope you have a good time with the beautiful minishell...

2.2. Bonus Pooh Oh Owners

Bonus parts are checked only when the required parts are perfect. "From start to finish"!

2.2.1. double left redirection <<

Make sure << works fine.

2.2.2. Line editing

0 (fail) to 5 (excellent)
- Does pressing **left** or **right** to move, insertion and deletion work well?
- Does the paste work well wherever I use it?
- Move by word by pressing **ctrl left** , **ctrl right** ?
- Move to the beginning or end of a sentence by pressing **home** , **end** ?
- Does it work across multiple lines?

2.2.3. And, Or &&, ||

0 (fail) to 5 (excellent)
- Make sure using && and || works like bash.
- 1 point for every flag that works!
- 1 bonus if all flags work!

2.2.4. Wildcard Wildcard

- Use wildcards in parameters.
- Try using */*.