Assumptions

TABLE-OF-CONTENTS:

1. Applications assumptions

There are some assumptions we made in our implementation, we demonstrate them here with a few examples. Some applications are not included here since they have clear functional descriptions.

General:

1. File names containing dash (-)

Assumptions:

We do not deal with files whose filenames contain dash (-) (For instance, -d.txt). In Unix-like system, a double dash (--) is used to signify the end of command options. Since we are not asked to implement this functionality, we regard such filenames to be invalid.

2. Newline at the end of application output

Assumptions:

For some of the applications, an extra newline is added at the end of their output, they are implemented in the original source code. Since we want to follow the standard and style of the original code, we choose to keep the newline in our implementation. However, for applications like **paste**, it's reasonable to remove the newline to match the expected behaviour stated in the project description, so we removed it in such cases.

3. Command with no arguments

Assumptions:

Some commands, such as sort and wc, would take the standard input stream as input. We do not allow these commands to take the standard input stream without IO Redirection or Pipe. And the error message "insufficient arguments" would be thrown.

4. File permission

Assumptions:

According to discussion in lab, we do not consider the file permission when implementing functionalities.

THE BASIC FUNCTIONALITIES:

1.1. RM

Assumptions:

- Assume the flags do not have scope, which means rm folder1 -r folder2 is the same as rm -r folder1 folder2. This is different with linux.
- Rm do not guarantee atomicity. For example, rm file1 file2 file3, if file3 can not be removed due to some reasons, file1 and file2 still will be removed. But for rm file3 file1 file2, this will not remove any file since it will be interrupted in running rm file3.

1.2. ECHO

Assumptions:

- It will print corresponding messages with a new line character in the end.
- In case of command substitution, it will replace newline to whitespace. For example, echo "`echo hello`world" will output hello world. The additional whitespace is changed from the new line.

Command format

echo [message]

Example

echo "hello world" ⇒ hello world

1.3. PASTE

Assumptions

- no new line adds to the end of the result, no \t add to the end of a line.
- Exception is thrown immediately after one invalid file name is found.
- The paste command can also be used to merge N consecutive lines from a file into a single line. Here N can be specified by specifying number hyphens(-) after paste.

Command format

paste [FILE] ...

FILE – the name of the file or files. If not specified, use stdin.

Examples:

```
# Merge stdin and two files A.txt and B.txt
$ paste A.txt - B.txt
```

A.txt	stdin	B.txt	output
1	A	1	1 A 1
2	B	3	2 B 3
3	C	5	3 C 5
4	D	7	4 D 7

1.4. SED

Assumptions

- an empty regular expression is considered as invalid, an exception would be thrown.

1.5. EXIT

Assumptions

- The EXIT application would call `System.exit(0)` immediately instead of break loop in `main()`.

THE EXTENDED FUNCTIONALITIES 1

1.6. GLOBBING

Assumptions

- The symbol * (asterisk) can only appear in the end file name. For example, our shell accept path like ".\testFiles\test*", but does not accept ".*\test1.txt"

1.7. DIFF

Assumptions

- While comparing two text files, we always find the longest common sublist of lines of two files. So, In this example, there are two common sublist candidates: line1 line3 line5 and line1 line4. As we always keep the longest common sublist, line1 line3 line5 is chosen.
  For example: diff A.txt B.txt

A.txt	B.txt	Output
line1	line 1	<line2
line2	line 3	<line4
line4	line 5	>line4
line3	line 4	>line6
line5	line 6	

- While comparing two directories, no matter which format of path is used, we always only use the pure folder name.
  For example:

  `$ diff A B`, `$ diff ./A ./B` or `$ diff /Users/user/path/A /Users/user/path/B` should get exactly the same results:

```
Common subdirectories: A/old and B/old
Only in A: out_image2.bmp
diff A/createAcct.sh B/createAcct.sh
< for u in $ul
> for u in $ul1
```

- If the `diff folder file.txt` is executed, a DiffException with message "Can not compare file with folder" will be thrown.
- While `diff` between stdin with file, no matter what arg order is given, the order is always diff file -
- While comparing two directories, if two files have the same name, but one is a regular file and one is a directory. The shell would output "File A/dir is a directory while file B/dir is a regular file", and we do not care whether the regular file is empty or not like the Unix shell does.
- We do not support comparing a binary file with stdin, and a DiffException "Can not compare binary file with input stream" would be thrown.

1.8. GREP

1.9. WC

Assumptions

- We do not care about the order of the flags.
- The WcApplication would always output an indent `\t` in front of every line, and output a `STRING_NEWLINE` at the end of every line.

1.10. CD

Assumptions

- if we get a cd command without other argument, like "cd ", our shell will not move to the root directory, instead, it will stay in the current directory.

1.11. CP

Assumptions

- We don't use the ending "/" to distinguish whether target is a file or directory.
- We assume that if target does not exist, then we will copy source to the target.
- Only if the target existed and is a directory, we will copy the file into this directory.
- If the target is an existing filem we will throw exception.

Noticed: In such case, definitely, we will not allow to copy a file to itself.

THE EXTENDED FUNCTIONALITIES 2

1.12. CUT

Assumptions

- the application can take a list of two numbers separated by comma, a range of numbers or a single number.
- If the number is out of range of the line's length, an exception will be thrown.
- two numbers separated by comma may have the first number greater than the second number, the cut result would be in the same order of the two number
- If the input range has the start number greater than the end number, an exception will be thrown.

Command format

Example

```
# Throw Out Of Range exception
$ echo "baz" | cut -b 8
# Display 'sT'. Suppose the file contains one line: "Today is Tuesday."
$ cut -c 8,1 test.txt
# Throw Invalid Range exception
$ cut -c 8-1 test.txt
```

1.13. LS

Assumptions :

- While multiple arguments are given to Ls application, the output for every arguments will be separated by STRING_NEWLINE.
  For example:
  ls file1 file 2
  Output:
  file1

  file2

1.14. SORT

1.15. FIND

Assumptions :

- Our find application only searches in the given relative path to the current working directory.

1.16. MV

Assumptions :

By default, it will overwrite an existing file. With the "-n" flag, it will not overwrite any existing file.

How to distinguish moving a file (including the folder) to the new file or moving files to the target folder?

- ***If the target does not exist, definitely moving a file to the new file***
- ***If the target exists and is a directory, then moving the files into the folder***
- ***If the target is a existing file,***
  - ***without -n, replace it***
  - ***with -n, throw execption.***

Command format:

mv [-n] SOURCE TARGET
mv [-n] [SOURCE] ... DIRECTORY

Example

mv file1.txt folder1

Noticed: It will throw exception if trying to move a file into itself (e.g. cp test.txt test.txt; cp test.txt ./).