Assumptions

General

- File names should not contain dash ‘`-`’. since it is used as an indicator for no files in some commands, `-m.txt` for example.
- When given multiple invalid flags in the command, only the first invalid flag will be flagged out.
- The exit status code serves no meaning in this shell application.
- Any valid flags across all applications should be used as a file or directory name. For example, `-n`, which is a valid flag in Cat application, should not be used as a directory/file name.
- If a string dash is used to specify no files, it should only be specified once, and not more than once.

Application Commands

Echo

- For an empty echo command, a new line will still be printed.

Cd

- For an empty cd command, ‘missing argument’ error will be thrown, unlike the behavior in MacOS’s terminal where the directory will be changed to the user’s home directory.
- Calling `cd` with more than 1 argument will throw an error.
- Only can be used on executable folders.

Cat

- Assumes that the list of files all have read and write permissions.
- Any empty lines will be regarded as lines in the file, and will still be prefixed with the line numbers with the `-n` flag.

Wc

- File names cannot be “-” or “<” as they will be parsed as tokens for indicating standard input.
- ‘-’ will not be appended to the results as it is considered as standard input, not file name.
- Can only take in standard input by parsing a ‘.txt’ file.
  - Example: `wc  test.txt - < input .txt`
- Similar to behavior as described in the project document.
- Users can only supply the c, l, w flag.

Mkdir

- The usage of -p will cause all the directories following it to create a directory along with its parent folders.

Sort

- The output of running sort is similar to MacOS’s terminal.
- Users can only supply the n, r, f flag.

Exit

- Any subsequent arguments after the `exit` keyword will be ignored and the application will proceed to exit with the status code of 0.

Ls

- If an error occurs while listing multiple directories due to an invalid directory, ls will continue showing output for the remaining directories

- Show folders when there are more than one folder

Paste
- Paste requires read permission
- Paste will raise an exception without executing if the given source is invalid, such as when a source is a directory or when there's a denial of source permission.
- Provided no exceptions occur, the output will have as many columns as the number of provided sources.

Uniq
- Using the -c and -D flags together will lead to an error, mirroring the behavior observed in bash.

Mv
- The output mimics that of the shell mv command when employing the -v flag.
- When executing the mv command with several source files, if any file causes an exception, the command processes and moves all source files listed before the faulty one successfully, it does not attempt to move any files listed after the faulty one.

Cut
- Repeated positions will only be processed once. For example, `cut -b 1,1,1-1`\`is equivalent to `cut -b 1`.
- The output will always be displayed in order of ascending positions. Given that the input stream is "`Hello`", `cut -b 5,2-4 -` will return `Ello`.
- Floats and negative numbers fed as arguments will be deemed to be invalid.
- If the input range has a the first number greater than the second number, `cut -b 5-1`, an exception will be thrown
- If there is an overflow for the end index of the range, no exception is thrown, it returns up to the point of EOF, just like the behavior in bash. For example, running `cut -b 1-100000` on an input stream of `Hello`, returns `Hello`.
- Comma separated numbers should not have a whitespace in between them (e.g. `5, 6`)

Rm
- Not possible to remove the current directory '`.`' or any directory ending in '`/.`' or '`/..`'
- No rollbacks for file deletions upon encountering an exception with deleting one of the multiple specified files. For example, in `rm file1.txt file2.txt file3.txt`, if an exception occurs for any reasons when attempting to remove `file2.txt`, the deletion of `file1.txt` will not be rolled back, whereas deletion of `file3.txt` will also not happen since an exception is already thrown and files are deleted in order.

Tee
- Appending content will add a newline, ensuring it's appended on a new line even if the last line doesn't end with one. (For example, if the file content is "Hello" or "Hello\n", and "World" is read in from standard input, the final file content will be "Hello\nWorld" in both cases.)
- Writing or appending output to a file won't automatically add a newline at the end of the file.

Grep
- If file does not exist, grep will continue showing output for other files
- Only one input stream allowed in one command, also only first dash will be processed, subsequent dashes will be ignored

Shell Operators

Call Command

-

Pipe Command

- For a pipe command, if any part of the pipe command is invalid, the whole command is deemed as invalid. For example, in `echo test | invalid | echo test`, it terminates and throws an invalid app message.

Sequence Command

- For a sequence command, if the first command is invalid and the second is valid, the error message will be displayed for the first command, while the second command will still execute. For example in `invalid; ls`, shell will throw an invalid app exception, followed by the output of the `ls` command.
- A command cannot begin with a sequence operator.

Globbing

- Do not support two asterisks in a single argument e.g.. src*/*

Quoting

- If any quotes in a command are unmatched, it will throw a syntax error.

IO-Redirection

- Redirecting both input and output to/from the same file should not be used.
- IORedirection to an output file will assume that the output file has read and write permissions.
- Likewise, IORedirection from an input file will assume that the input file has read and write permissions.

Command Substitution

- echo \`exit\` is a valid command that will cause the program to exit automatically when command substitution occurs
- For nested command substitutions, the commands are evaluated from “inside-out”. For example, in ``echo `echo `echo test` ` ``, the most inner `echo test` is evaluated first.