========== CAT ==========

Before running following commands, change directory to where the test files reside in:
cd testFiles/cat

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: Cat without arguments
Resource Files: None
Input: cat
Expected behaviour: Read from standard input and echo every input line until EOF (ctrl-D).

Description: Cat with '-' argument
Resource Files: None
Input: cat -
Expected behaviour: Read from standard input and echo every input line until EOF (ctrl-D).

Description: Cat without arguments with line number
Resource Files: None
Input: cat -n
Expected behaviour: Read from standard input and echo every input line, together with line number, until EOF (ctrl-D).

Description: Cat with '-' argument with line number
Resource Files: None
Input: cat -n -
Expected behaviour: Read from standard input and echo every input line, together with line number, until EOF (ctrl-D).

Description: Cat an empty file with line number
Resource Files: emptyFile.txt
Input: cat -n emptyFile.txt
Expected behaviour: No output. No line number should be printed.

Description: Cat one file with -n option.
Resource Files: fileOne.txt
Input: cat -n fileOne.txt
Expected behaviour: Outputs as follows
1 I love
2 CS4218
3

Description: Cat one file full of special characters and symbols with line number
Resource Files: fileThree.txt
Input: cat -n fileThree.txt
Expected behaviour: Outputs as follows
1 " ' < > * | ` "
2 \ $ % ; , . ? /

Description: Cat multiple repeated files
Resource Files: fileTwo.txt
Input: cat fileTwo.txt fileTwo.txt
Expected behaviour: Outputs as follows
~!@#$%+
~!@#$%+

Description: Cat multiple files without line number
Resource Files: fileOne.txt, fileTwo.txt, fileThree.txt
Input: cat fileOne.txt fileTwo.txt fileThree.txt
Expected behaviour: Outputs as follows
I love
CS4218

~!@#$%+
" ' < > * | ` "
\ $ % ; , . ? /

Description: Cat multiple files with line number
Resource Files: fileOne.txt, fileTwo.txt, fileThree.txt
Input: cat -n fileOne.txt fileTwo.txt fileThree.txt
Expected behaviour: Outputs as follows
1 I love
2 CS4218
3
4 ~!@#$%+
5 " ' < > * | ` "
6 \ $ % ; , . ? /

Description: Cat multiple files but one file is a directory, with line number
Resource Files: fileOne.txt, myDirectory, fileThree.txt
Input: cat -n fileOne.txt myDirectory fileThree.txt

Expected behaviour: Outputs as follows, error message should not take up a line number
1 I love
2 CS4218
3
cat: This is a directory
4 " ' < > * | ` "
5 \ $ % ; , . ? /

Description: Cat multiple files but one file is non-existing, with line number
Resource Files: fileOne.txt, fileTwo.txt
Input: cat -n nonExisting.file fileOne.txt fileTwo.txt
Expected behaviour: Outputs as follows, error message should not take up a line number
cat: No such file or directory
1 I love
2 CS4218
3
4 ~!@#$%+

Description: Cat one file and one '-' argument, with line number
Resource Files: fileTwo.txt
Input: cat -n - fileTwo.txt
Expected behaviour: Read from standard input and echo every input line, together with line number, until EOF (ctrl-D). After inputting x lines, output as follows.
^D
x+1 ~!@#$%+

Description: Cat multiple files with multiple '-' arguments, without line number
Resource Files: fileOne.txt fileTwo.txt
Input: cat - fileOne.txt - fileTwo.txt
Expected behaviour: Read from standard input and echo every input line until EOF (ctrl-D). Then output as follows:
^D
I love
CS4218

~!@#$%+

Description: Cat multiple files with multiple '-' arguments, but one file is non-existing and one file is a directory, with line number
Resource Files: fileOne.txt, fileTwo.txt myDirectory

Input: cat -n fileTwo.txt - nonExisting.baa - myDirectory fileOne.txt
Expected behaviour: First output as follows,
1 ~!@#$%+
then read from standard input and echo every input line until EOF (ctrl-D). After inputting x lines, output as follows:
^D
cat: No such file or directory
cat: This is a directory
x+2 I love
x+3 CS4218
x+4

NEGATIVE

Description: Cat one non existing file
Resource Files:
Input: cat -n nonExisting.file
Expected behaviour: Output as follows
cat: No such file or directory

Description: Cat one valid file but with capital -n option
Resource Files: fileOne.txt
Input: cat -N fileOne.txt
Expected behaviour: Output as follows
cat: Invalid flag option supplied

Description: Cat one valid file but with invalid option
Resource Files: fileOne.txt
Input: cat -r fileOne.txt
Expected behaviour: Output as follows
cat: Invalid flag option supplied

Description: Cat one valid file but with two -n options
Resource Files: fileOne.txt
Input: cat -n -n fileOne.txt
Expected behaviour: Output as follows. The second -n is regarded as a file, which does not exist.
cat: No such file or directory
1 I love
2 CS4218

3

Description: Cat one valid directory
Resource Files: myDirectory
Input: cat myDirectory
Expected behaviour: Output as follows
cat: This is a directory

POSITIVE INTEGRATION with (quoting, IO, command sub, semicolon, globbing, pipe)

Description: Cat with quoting
Resource Files: fileOne.txt fileTwo.txt myDirectory
Input: cat -n "fileTwo.txt" nonExisting.file 'myDirectory' fileOne.txt
Expected behaviour: Output as follows
1 ~!@#$%+
cat: No such file or directory
cat: This is a directory
2 I love
3 CS4218
4

Description: Cat with IO redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: cat -n fileTwo.txt nonExisting.file fileOne.txt - < fileThree.txt > output.txt
Expected behaviour: output.txt file created. Contents as follows. (output file may not appear on intelliJ immediately)
1 ~!@#$%+
cat: No such file or directory
2 I love
3 CS4218
4
5 " ' < > * | ` "
6 \ $ % ; , . ? /

Description: Cat with semicolon.
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: cat -n fileTwo.txt nonExisting.file fileThree.txt myDirectory fileOne.txt; echo hello
Expected behaviour: Output as follows. echo does not take up a line number from cat.
1 ~!@#$%+
cat: No such file or directory

2 " ' < > * | ` "
3 \ $ % ; , . ? /
cat: This is a directory
4 I love
5 CS4218
6
hello

Description: Cat with globbing
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: cat -n *.txt nonExisting.file myDirectory fileOne.txt
Expected behaviour: Output as follows. Glob arranges .txt in alphabetical order: fileOne.txt, fileThree.txt, then fileTwo.txt
1 I love
2 CS4218
3
4 " ' < > * | ` "
5 \ $ % ; , . ? /
6 ~!@#$%+
cat: No such file or directory
cat: This is a directory
7 I love
8 CS4218
9

Description: Cat with pipe and grep
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: cat -n fileOne.txt fileTwo.txt fileThree.txt nonExisting.file | grep %
Expected behaviour:
4 ~!@#$%+
6 \ $ % ; , . ? /

Description: Cat with quote, IO redirection, pipe, globbing and semicolon, with non-existing and directory files
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: cat -n *.txt "nonExisting.file" myDirectory 'fileOne.txt' - < fileThree.txt | grep 1; echo hello
1 I love
2 CS4218
8 CS4218
10 " ' < > * | ` "
11 \ $ % ; , . ? /

hello

NEGATIVE INTEGRATION

Description: Cat with IO redirection but multiple files provided for input redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: cat -n fileTwo.txt nonExisting.file - < fileThree.txt fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: Cat with IO redirection but multiple input redirections
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: cat -n fileTwo.txt - < fileThree.txt - < fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: Valid cat arguments but all quoted as one
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: cat -n "fileTwo.txt fileThree.txt fileOne.txt"
Expected behaviour: Output as follows
cat: No such file or directory

========== CD ==========

Please run the following commands in the project root directory.

POSITIVE

Description: cd without argument
Resource Files: None
Input: cd
Expected behaviour: Current directory remains unchanged.

Description: cd with single dot
Resource Files: None
Input: cd .
Expected behaviour: Current directory remains unchanged.

Description: cd with double dot
Resource Files: None
Input: cd ..

Expected behaviour: Current directory changed to project root's parent directory

Description: cd with single dot then double dot
Resource Files: None
Input: cd ./..
Expected behaviour: Current directory changed to project root's parent directory

Description: cd with double, single, double dots, ending with a slash
Resource Files: None
Input: cd .././../
Expected behaviour: Current directory changed to project root's grandparent directory

Description: cd into a directory
Resource Files: src
Input: cd src
Expected behaviour: Current directory changed to src

Description: cd into a directory with absolute path
Resource Files: src
Input: cd <absolute path of 'src' directory>
Expected behaviour: Current directory changed to src

Description: cd into a directory, then double dots and into another directory
Resource Files: src
Input: cd src/../testFiles
Expected behaviour: Current directory changed to testFiles

Description: cd into an inner directory
Resource Files: src/sg/
Input: cd src/sg/
Expected behaviour: Current directory changed to src/sg

Description: cd into an inner directory but with redundant slashes
Resource Files: src/sg/
Input: cd //src/////sg/////
Expected behaviour: Current directory changed to src/sg

Description: cd into an inner directory but with redundant slashes and dots
Resource Files: src/sg/
Input: cd src///.///../src//sg/.
Expected behaviour: Current directory changed to src/sg

NEGATIVE

Description: cd but two arguments given
Resource Files: src test
Input: cd src test
Expected behaviour: Output as follows:
cd: Too many arguments

Description: cd into non existing directory
Resource Files: None
Input: cd nonExistingDir
Expected behaviour: Output as follows:
cd: No such file or directory

Description: cd into a file
Resource Files: testFiles/cat/fileOne.txt
Input: cd testFiles/cat/fileOne.txt
Expected behaviour: Output as follows.
cd: Not a directory

Description: cd into a non-existing directory at the middle of path
Resource Files: src
Input: cd src/nonExistingDir/..
Expected behaviour: Output as follows.
cd: No such file or directory

Description: cd with three dots
Resource Files: None
Input: cd .../
Expected behaviour: Output as follows.
cd: No such file or directory

Description: cd into '~'
Resource Files: None
Input: cd ~
Expected behaviour: Output as follows. Assume windows OS. No home directory (i.e. ~)
cd: No such file or directory

POSITIVE INTEGRATION (with quoting, IO, command sub, semicolon, globbing, pipe)

Description: cd with double quoting
Resource Files: src
Input: cd "src"

Expected behaviour: Current directory changed to src

Description: cd with single quoting
Resource Files: src
Input: cd 'src'
Expected behaviour: Current directory changed to src

Description: cd with IO redirection
Resource Files: src
Input: cd src < testFiles/cat/fileOne.txt > output
Expected behaviour: Current directory changed to src. Empty 'output' file created. As per UNIX behavior.

Description: cd with input redirection
Resource Files: README.md
Input: cd < README.md
Expected behaviour: Current directory remains unchanged. Unix would redirect to home directory. But since we do not have home directory, it is a no-op.

Description: cd with command substitution
Resource Files: src
Input: cd "`echo src`"
Expected behaviour: Current directory changed to src.

Description: cd with semicolon
Resource Files: src/sg
Input: cd src; cd sg;
Expected behaviour: Current directory changed to src/sg

Description: cd with globbing, but only matched with one directory
Resource Files: src
Input: cd sr*
Expected behaviour: Current directory changed to src

NEGATIVE INTEGRATION

Description: cd with globbing, but matched with multiple directories
Resource Files: test, testFiles
Input: cd tes*
Expected behaviour: Output as follows
cd: Too many arguments

Description: cd with command substition, but disabled by single quotes

Resource Files: None
Input: cd '`echo src`'
Expected behaviour: Output as follows
cd: No such file or directory

========== ECHO ==========

Please run the following commands in the project root directory.

POSITIVE

Description: echo with words and numbers
Resource Files: src
Input: echo "cool231"
Expected behaviour: cool231

Description: echo with symbols
Resource Files: src
Input: echo "%*R#"
Expected behaviour: %*R#

Description: echo with empty string
Resource Files: src
Input: echo ""
Expected behaviour: A new line

Description: echo words and pass into wc
Resource Files: src
Input: echo "cool like this" | wc -w
Expected behaviour: Output 3

Description: Pipe echo into echo
Resource Files: src
Input: echo "hey" | echo "echooo"
Expected behaviour: output “echooo”

NEGATIVE

Description: echo words and pass into wc
Resource Files: src
Input: echo "

Expected behaviour: Output “shell: Invalid syntax”

========== LS ==========

Please run the following commands in the testFolders/app/lsTestFolder/ directory.

POSITIVE

Description: ls with only directories
Resource Files: testFolders/app/lsTestFolder/
Input: ls -d
Expected behaviour:
ls -d
dir
dir1
dir2

Description: ls with option X
Resource Files: testFolders/app/lsTestFolder/
Input: ls -X
Expected behaviour:
dir
dir1
dir2
TEST1.md
test2.txt
test3.xml

Description: ls with option d and X
Resource Files: testFolders/app/lsTestFolder/
Input: ls -X
Expected behaviour:
ls -dX
dir
dir1
dir2

Description: Pipe cd with ls with grep
Resource Files: testFolders/app/lsTestFolder/
Input:cd pipeFolderTest/ | ls | grep ".txt" | grep "Cat"
Expected behaviour: Output “CatGrepTest.txt”

Description: ls with globbing
Resource Files: testFolders/app/lsTestFolder/
Input: ls *
Expected behaviour: All file contents will be listed out including contents of directories.

NEGATIVE

Description: ls test with illegal option
Resource Files: testFolders/app/lsTestFolder/
Input: ls -x
Expected behaviour: Output “ls: illegal option -- x”

========== MV ==========

Please run the following commands in the testFolders/app/mvTestFolder directory.

POSITIVE

Description: Rename dir4 to dir5
Resource Files: testFolders/app/mvTestFolder
Input: mv dir4 dir5
Expected behaviour: dir4 will be renamed to dir5

Description: Rename a file
Resource Files: testFolders/app/mvTest/Folder
Input: mv beforeRename.txt afterRename.txt
Expected behaviour: beforeRename.txt will be renamed to afterRename.txt

Description: Rename afterRename.txt to file1.txt with overwriting allowed
Resource Files: testFolders/app/mvTest/Folder
Input: mv afterRename.txt file1.txt
Expected behaviour: afterRename.txt will overwrite file1.txt

Description: Move file into dir
Resource Files: testFolders/app/mvTestFolder
Input: mv file1.txt dir1
Expected behaviour: file1.txt will be moved into dir1

Description: Move dir1 into dir3
Resource Files: testFolders/app/mvTestFolder
Input: mv dir1 dir3

Expected behaviour: dir1 will be moved into dir3

NEGATIVE

Description: Mv test with illegal option
Resource Files: testFolders/app/mvTestFolder
Input: mv -q dir5 dir4
Expected behaviour: Output “mv: illegal option -- q”

Description: Rename a non existing file
Resource Files: testFolders/app/mvTest/Folder
Input: mv notExistFile.txt random.txt
Expected behaviour: Output “mv: cs4218-project-ay2021-s2-2021-team22/mvFolderTest/notExistFile.txt not found. No such file or directory”

Description: Missing argument
Resource Files: testFolders/app/mvTestFolder
Input: mv beforeRename.txt
Expected behaviour: Output “mv: Missing Argument”

Description: Rename existing file names with no overwriting
Resource Files: testFolders/app/mvTestFolder
Input: mv -n afterRename.txt file1.txt
Expected behaviour: Output “mv: Target filename file1.txt already exists.”

Description: Too many arguments
Resource Files: testFolders/app/mvTestFolder
Input: mv file2.txt file1.txt file3.txt
Expected behaviour: Output “mv: Too many arguments”

========== GREP ==========

Before running following commands, change directory to where the test files reside in:
cd testFiles/grep

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: grep with pattern and no [FILES]
Resource Files: None

Input: grep e
Expected behaviour: Read from standard input and returns list of lines that matches PATTERN after CTRL + D

Description: grep with '-' argument
Resource Files: None
Input: grep e -
Expected behaviour: Read from standard input and returns list of lines that matches PATTERN after CTRL + D

Description: grep with 1 FILE but file is empty
Resource Files: emptyFile
Input: grep e emptyFile
Expected behaviour: Nothing. Literally expect nothing

Description: grep with 1 FILE
Resource Files: testFile1
Input: grep e testFile1
Expected behaviour: Reads content of FILE and returns list of lines that matches PATTERN
Output Follows
I really like

Description: grep with FILES
Resource Files: testFile1, testFile2
Input: grep e testFile1 testFile2
Expected behaviour: Reads content of FILES and returns list of lines that matches PATTERN
Output Follows
testFile1: I really like
testFile2: .~!@#$%+e

Description: grep with FILE and stdin
Resource Files: testFile1
Input: grep e - testFile1
Expected behaviour: Reads from standard input and content of FILE and returns a list of lines that matches PATTERN, each returned line is prefixed with either their source.
Output Follows:
(standard_input): …

testFile1: I really like

Description: grep -i flag and stdin
Resource Files:
Input: grep -i e

Expected behaviour: Reads from standard input and returns a list of lines that matches case insensitive matches to PATTERN

Description: grep repeated flag and stdin
Resource Files:
Input: grep -i -i e
Expected behaviour: Reads from standard input and returns a list of lines that matches case insensitive matches to PATTERN

Description: grep -c flag with stdin
Resource Files:
Input: grep -c e
Expected behaviour: Reads content from standard input returns a count of lines that matches PATTERN

Description: grep -H flag with stdin
Resource Files:
Input: grep -H e
Expected behaviour: Reads content from standard input and returns a count of lines that matches PATTERN prefixed by “(standard_input): ”

Description: grep all flags with stdin
Resource Files:
Input: grep -i -c -H e
Expected behaviour: Reads content from standard input and returns a count of lines that matches PATTERN. -c flag has precedence just like linux

Description: grep -d and -H flag with stdin
Resource Files:
Input: grep -d -H e
Expected behaviour: Reads content from standard input and returns a list of lines that matches case insensitive PATTERN prefixed by “(standard_input):”

Description: grep with file in sub directory
Resource Files: testDir/subFile1
Input: grep e testDir/subFile1
Expected behaviour: Reads content from file and returns a list of lines that matches PATTERN
OutputFollows
subFile1

Description: grep multiple files with one file as a directory and line count
Resource Files: testFile1, testDir

Input: grep e testFile1 testDir
Expected behaviour: error message doesn't return line count,
Outputs as follows:
testDir: Is a directory
testFile1: I really like

Description: grep multiple files with one FILE not existing and line count
Resource Files: testFile1
Input: grep e -H testFile1 nonExist
Expected behaviour: error message doesn't return line count,
Outputs as follows:
testFile1: 1

nonExist: No such file or directory

NEGATIVE

Description: grep non existing file
Resource Files:
Input: grep e nonExist
Expected behaviour: error message output,
Outputs as follows:
nonExist: No such file or directory

Description: grep non recognized flag with file
Resource Files: testFile1, PATTERN = "e"
Input: grep -A e
Expected behaviour: error message output, - recognized as pattern, and e as filename
Outputs as follows:
e: No such file or directory

Description: grep no pattern
Resource Files:
Input: grep
Expected behaviour: error message returned
Output Follows

Description: grep with 2 '-'
Resource Files:
Input: grep e - -
Expected behaviour: error message returned
Output Follows
Grep: Stream is closed

POSITIVE INTEGRATION with (quoting, IO, command sub, semicolon, globbing, pipe)

Description: grep with quoting
Resource Files: testFile1
Input: grep “e” testFile1
Expected behaviour: same as grep e testFile1

Description: grep with IO redirection output
Resource Files: testFile1
Input: grep e - testFile1 > output
Expected behaviour: Line matches in testFile1 with PATTERN written to output

Description: grep with IO redirection input
Resource Files:
Input: grep e - < testFile1
Expected behaviour: same as grep e testFile1

Description: grep with semicolon.
Resource Files: testFile1
Input: grep -c e testFile1; echo hello
Expected behaviour: Output as follows. echo output printed after grep e testFile1.
1
Hello

Description: grep with globbing.
Resource Files: testFile1 testFile2
Input: grep -c testFile*
Expected behaviour: Output as follows. Glob arranges .txt in alphabetical order: testFile1 testFile2

1
1

Description: grep with pipe and cat
Resource Files: testFile1
Input: grep e -c testFile1 | cat

Expected behaviour:
1

Description: grep with quote, IO redirection, pipe, globbing and semicolon, with non-existing and directory files
Resource Files: testFile1 testFile2
Input: grep -c “e” testFile* 'emptyFile' - < testDir/subFile1 | cat; echo hello

1
1
2
hello

NEGATIVE INTEGRATION

Description: Grep with IO redirection but multiple files provided for input redirection
Resource Files: testFile1 testFile2
Input: grep e < testFile1 testFile2 > output
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: grep with IO redirection but multiple input redirections
Resource Files: testFile1 testFile2
Input: grep e - < testFile1 > output > output2
Expected behaviour: Output as follows
shell: Multiple Streams Provided

Description: Files bunched in quote
Resource Files: testFile1 testFile2
Input: grep -e "testFile1 testFile2"
Expected behaviour: Output as follows
testFile1 testFile2: No such file or directory

========== SPLIT ==========

Before running following commands, change directory to where the test files reside in:
cd testFiles/split

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: split with stdin
Resource Files:
Input: split
Expected behaviour: standard in by user is split by default 1000 lines into xaa. If standard input exceeds 1000 lines input is split into xab and so on. If more than 676000 lines of inputs are given, the 676001th line will be split into zaa.

Description: split with FILE with prefix

Resource Files: defaultFile
Input: split defaultFile prefix
Expected behaviour: content of file is split by default 1000 lines into prefixaa. If file lines exceeds 1000 lines input is split into xaa and so on.

Description: split FILE by x lines
Resource Files: testFile1
Input: split -l 2 testFile1
Expected behaviour: content of file is split for every 2 lines into xaa. If file lines exceeds 676 * 2 lines input is split into zaa and so on.
xaa :
test1
test2

xab:
test3

Description: split FILE by b bytes
Resource Files: testFile1
Input: split -b 10 testFile1
Expected behaviour: content of file is split for every 10 bytes into xaa. If file lines exceeds 676 * 10 bytes input is split into zaa and so on.

Description: split FILE by b bytes
Resource Files: testFile1
Input: split -b 10b testFile1
Expected behaviour: content of file is split for every 10 * 512 bytes into xaa. If file lines exceeds 676 * 10 * 512 bytes input is split into zaa and so on.

Description: split FILE by b bytes
Resource Files: testFile1
Input: split -b 10k testFile1
Expected behaviour: content of file is split for every 10 * 1024 bytes into xaa. If file lines exceeds 676 * 10 *1024 bytes input is split into zaa and so on.

Description: split FILE by b bytes
Resource Files: testFile1
Input: split -b 10m testFile1
Expected behaviour: content of file is split for every 10 * 1048576 bytes into xaa. If file lines exceeds 676 * 10 * 1048576 bytes input is split into zaa and so on.

NEGATIVE
Description: split FILE by 0 lines
Resource Files: testFile1

Input: split -l 0 testFile1
Expected behaviour: Returns error message
Outputs follow:
split: Invalid lines count

Description: split FILE by -ve lines
Resource Files: testFile1
Input: split -l -ve testFile1
Expected behaviour: Returns error message
Outputs follow:
split: Invalid lines count

Description: split FILE by 0 bytes
Resource Files: testFile1
Input: split -b 0 testFile1
Expected behaviour: Returns error message
Outputs follow:
split: Invalid bytes count

Description: split FILE by -ve bytes
Resource Files: testFile1
Input: split -b -200 testFile1
Expected behaviour: Returns error message
Outputs follow:
split: Invalid bytes count

Description: split FILE by lines repeated flags
Resource Files: testFile1
Input: split -l 2 -l 2 testFile1
Expected behaviour: second time flag is read, it is read as a file. Returns error message
Outputs follow:
split: Invalid Syntax

Description: split FILE by lines multi flags
Resource Files: testFile1
Input: split -l 2 -b 2 testFile1
Expected behaviour: second flag read is read as file. Returns error message
Outputs follow:
split: Invalid Syntax

Description: split FILE by lines multi prefix provided
Resource Files: testFile1
Input: split -l 2 -b 2 testFile1 prefix1 prefix2
Expected behaviour: Returns error message

Outputs follow:
split: Invalid Syntax

POSITIVE INTEGRATION(with quoting, IO, command sub, semicolon, globbing, pipe)

Description: split with quoting
Resource Files: testFile1
Input: split “testFile1”

Expected behaviour: same as split testFile1
Description: split with input redirection
Resource Files: testFile1
Input: split < testFile1
Expected behaviour: same as split testFile1

Description: split with globbing returns list of size 2
Resource Files: testFile1 testFile2(exists in directory)
Input: split testFile*
Expected behaviour: testFile2 read as prefix

Description: split with semicolon
Resource Files: testFile1
Input: split testFile;
Expected behaviour: same as split testFile1

Description: split with IORedirection
Resource Files: testFile1
Input: split < testFile1 > output
Expected behaviour: same as splitFile1 but empty ‘output’ file created.

NEGATIVE INTEGRATION

Description: split with globbing returns list of size 3
Resource Files: testFile1 testFile2(exists in directory) testFile3(exist in directory)
Input: split testFile*
Expected behaviour:
Outputs follow:
Split: Invalid Syntax

Description: split with IO redirection but multiple files provided for input redirection
Resource Files: testFile1 testFile2 (as prefix)
Input: split < testFile1 testFile2 > output
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: split with IO redirection but multiple input redirections
Resource Files: testFile1 testFile2
Input: split < testFile1 > output > output2
Expected behaviour: Output as follows
shell: Multiple Streams Provided

Description: split FILE and prefix bunched in quote
Resource Files: testFile1
Input: split "testFile1 prefix"
Expected behaviour: Output as follows
testFile1 testFile2: No such file or directory

========== UNIQ ==========
Before running following commands, change directory to where the test files reside in:
cd testFiles/uniq

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: Uniq without arguments
Resource Files: None
Input: uniq
Expected behaviour: shell echoes each line input by user. Same behaviour as linux.

Description: Uniq with input file given
Resource Files: testFile1
Input: uniq testFile1
Expected behaviour: Adjacent matching lines merged to first occurrence
Output Follows
I love CS4218
Andrew
Evon
Andrew
Evon
Ev0n

Description: Uniq with input file given and c flag
Resource Files: testFile1
Input: uniq -c testFile1
Expected behaviour: adjacent matching lines merged to first occurrence with count of adjacent lines prefixed

Output Follows
3 I love CS4218
2 Andrew
1 Evon
1 Andrew
1 Evon
1 Ev0n

Description: Uniq with input file and d flag
Resource Files: testFile1
Input: uniq -d testFile1
Expected behaviour: Only duplicate adjacent lines printed
Output Follows
I love CS4218
Andrew

Description: Uniq with input file and D flag
Resource Files: testFile1
Input: uniq -D testFile1
Expected behaviour: All duplicate lines printed
Output Follows
I love CS4218
I love CS4218
I love CS4218
Andrew
Andrew

Description: Uniq with input file and both d flags
Resource Files: testFile1
Input: uniq -d -D testFile1
Expected behaviour: All duplicate lines printed
Output Follows
I love CS4218
I love CS4218
I love CS4218
Andrew
Andrew

Description: Uniq with input file and output file
Resource Files: testFile1
Input: uniq -D testFile1 output.txt
Expected behaviour: All duplicate lines written to output.txt

Description: Uniq with input file and output file repeated flag

Resource Files: testFile1
Input: uniq -D -D testFile1 output.txt
Expected behaviour: All duplicate lines written to output.txt

Description: Uniq with input file and output file in directory
Resource Files: testFile1
Input: uniq -D testFile1 testDir/output
Expected behaviour: All duplicate lines written to testDir/output.txt

NEGATIVE

Description: Uniq with c and d flag
Resource Files: testFile1
Input: uniq -c -d testFile1
Expected behaviour: Error message returned as that combination is not meaningful. As described in luminus
Output Follows
Uniq: Invalid Syntax

Description: Uniq with input file as directory
Resource Files: testDir
Input: uniq testDir
Expected behaviour: Error Message returned
Output Follows
testDir: Is a directory

Description: Uniq with output file as directory
Resource Files: testFile1
Input: uniq testFile1 testDir
Expected behaviour: Error Message returned
Output Follows
testDir: Is a directory

Description: Uniq with non existent file as input
Resource Files:
Input: uniq nonExist
Expected behaviour: Error Message returned
Output Follows
testDir: No such file or directory

POSITIVE INTEGRATION(with quoting, IO, command sub, semicolon, globbing, pipe)

Description: uniq with quoting
Resource Files: testFile1
Input: uniq “testFile1”

Expected behaviour: same as uniq testFile1
Description: uniq with input redirection
Resource Files: testFile1
Input: uniq < testFile1
Expected behaviour: same as uniq testFile1

Description: uniq with globbing returns list of size 2
Resource Files: testFile1 testFile2(exists in directory)
Input: uniq testFile*
Expected behaviour: uniqFile2 read as outputFilename

Description: uniq with semicolon
Resource Files: testFile1
Input: uniq testFile;
Expected behaviour: same as uniq testFile1

Description: uniq with IORedirection
Resource Files: testFile1
Input: uniq < testFile1 > output
Expected behaviour: same as splitFile1 but empty ‘output’ file created.

NEGATIVE INTEGRATION

Description: uniq with globbing returns list of size 3
Resource Files: testFile1 testFile2(exists in directory) testFile3(exist in directory)
Input: uniq testFile*
Expected behaviour:
Outputs follow:
Split: Invalid Syntax

Description: uniq with IO redirection but multiple files provided for input redirection
Resource Files: testFile1
Input: uniq < testFile1 testfile2 > output
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: uniq with IO redirection but multiple output redirections
Resource Files: testFile1 testFile2
Input: split < testFile1 > output > output2

Expected behaviour: Output as follows
shell: Multiple Streams Provided

Description: uniq input and output file bunched in quotes
Resource Files: testFile1
Input: split "testFile1 output"
Expected behaviour: Output as follows
testFile1 output: No such file or directory

========== WC ==========

Before running following commands, change directory to where the test files reside in:
cd testFiles/wc

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: wc with stdin
Resource Files: -
Input: wc
Expected behaviour: displays the number of lines, words, and bytes for the user's input. The user's input will be terminated and processed when CTRL + D is hit.

Description: wc with -c flag with FILE
Resource Files: fileOne.txt
Input: wc -c fileOne.txt
Expected behaviour: displays the number of bytes for the content inside of the specified file, fileOne.txt Outputs as follows:

21 fileOne.txt

Description: wc with -l flag with FILE
Resource Files: fileOne.txt
Input: wc -l fileOne.txt
Expected behaviour: displays the number of lines for the content inside of the specified file, fileOne.txt Outputs as follows:

3 fileOne.txt

Description: wc with -w flag with FILE
Resource Files: fileOne.txt
Input: wc -w fileOne.txt

Expected behaviour: displays the words of lines for the content inside of the specified file, fileOne.txt. Outputs as follows:

```
5 fileOne.txt
```

Description: wc with -clw flag with multiple FILEs
Resource Files: fileOne fileTwo fileThree
Input: wc -c fileOne.txt fileTwo.txt
Expected behaviour: displays the number of lines, words, bytes for the content inside of the specified files, followed by the total lines, words, bytes for all the files. The output is as follows:

```
3      5     21 fileOne.txt
1      1      7 fileTwo.txt
3     17     46 fileThree.txt
7     23     74 total
```

Description: wc with -c flag with multiple FILEs
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc -c fileOne.txt fileTwo.txt fileThree.txt
Expected behaviour: displays the number of bytes for the content inside of the specified files, followed by the total bytes for all the files. The output is as follows:

```
21 fileOne.txt
 7 fileTwo.txt
46 fileThree.txt
74 total
```

Description: wc with -l flag with multiple FILEs
Resource Files: fileOne fileTwo fileThree
Input: wc -l fileOne.txt fileTwo.txt
Expected behaviour: displays the number of lines for the content inside of the specified files, followed by the total lines for all the files. The output is as follows:

```
3 fileOne.txt
1 fileTwo.txt
3 fileThree.txt
7 total
```

Description: wc with -w flag with multiple FILEs
Resource Files: fileOne fileTwo fileThree
Input: wc -c fileOne.txt fileTwo.txt
Expected behaviour: displays the number of words for the content inside of the specified files,, followed by the total words for all the files. The output is as follows:

```
 5 fileOne.txt
 1 fileTwo.txt
17 fileThree.txt
23 total
```

NEGATIVE

Description: wc a non existing file
Resource Files:
Input: wc nonExisting.file
Expected behaviour: Output as follows
wc: No such file or directory

Description: Cat one valid file but with invalid option
Resource Files: fileOne.txt
Input: wc -o fileOne.txt
Expected behaviour: Output as follows
wc: Invalid flag option supplied

Description: Cat one valid file but with invalid mixtures of flag options. Valid mixtures includes: -cl, -cw, -lc, -lw, -wc, -wl, -clw.
Resource Files: fileOne.txt
Input: wc -cg fileOne.txt
Expected behaviour: the entire flag is regarded as invalid, does not even display the byte of the file's content. Output as follows.
wc: Invalid flag option supplied

Description: wc one valid directory
Resource Files: myDirectory
Input: wcDirectory
Expected behaviour: Output as follows
wct: This is a directory

POSITIVE INTEGRATION with (quoting, IO, command sub, semicolon, globbing, pipe)

Description: wc with quoting
Resource Files: fileOne.txt fileTwo.txt myDirectory
Input: wc -cl "fileTwo.txt" NonExistingFile.txt 'myDirectory' fileThree.txt
Expected behaviour: Output as follows
    1   7 fileTwo.txt
wc: No such file or directory
wc: This is a directory
    3   46 fileThree.txt
    4   53 total

Description: wc with IO redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc -lw fileOne.txt nonExistingFile fileTwo.txt - < fileThree.txt > output.txt
Expected behaviour: output.txt file created. Contents as follows. (output file may not appear on intelliJ immediately)

```
      3       5 fileOne.txt
wc: No such file or directory
      1       1 fileTwo.txt
      3      17 -
      7      23 total
```

Description: wc with semicolon.
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: wc -c fileTwo.txt fileThree.txt myDirectory fileOne.txt; echo hello
Expected behaviour: Output as follows. Echo's result is after wc's results.

```
       7 fileTwo.txt
      46 fileThree.txt
wc: This is a directory
      21 fileOne.txt
      74 total
hello
```

Description: wc with globbing
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: wc -clw *.txt
Expected behaviour: Output as follows. Glob arranges .txt in alphabetical order: fileOne.txt, fileThree.txt, then fileTwo.txt

```
      3       5      21 fileOne.txt
      3      17      46 fileThree.txt
      1       1       7 fileTwo.txt
      7      23      74 total
```

Description: wc with pipe and tee
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc -c fileOne.txt fileTwo.txt fileThree.txt nonExisting.file | tee wcPipeTeeOutput.txt
Expected behaviour: the results from the wc operation will be written inside as the content of the new wcPipeTeeOutput.txt file. Both the output and content of the file is the same, as shown:

```
      21 fileOne.txt
       7 fileTwo.txt
      46 fileThree.txt
wc: No such file or directory
```

74 total

Description: wc with quote, IO redirection, pipe, globbing and semicolon, with non-existing and directory files
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: wc -cl *.txt "nonExistFile" myDirectory - < fileThree.txt | tee newOutput; echo hello
Expected behaviour: the wc operation is first executed before being written to a new file named newOutput. Then echo is executed.
Below is the Output:

```
      3      18 fileOne.txt
      3      46 fileThree.txt
      1       7 fileTwo.txt
wc: No such file or directory
wc: This is a directory
      3      46 -
     10     117 total
hello
```

NEGATIVE INTEGRATION

Description: wc with IO redirection but multiple files provided for input redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc fileTwo.txt nonExistingFile - < fileThree.txt fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: wc with IO redirection but multiple input redirections
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc -cl fileTwo.txt - < fileThree.txt - < fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: Valid wc arguments but all quoted as one
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: wc -clw "fileTwo.txt fileThree.txt fileOne.txt"
Expected behaviour: Output as follows
wc: No such file or directory

========== TEE ==========

Before running following commands, change directory to where the test files reside in:
cd testFiles/tee

Please remember to do so after every CTRL+D for reading standard input.

POSITIVE

Description: tee with stdin
Resource Files: -
Input: tee
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out.

Description: tee with FILE
Resource Files: fileOne.txt
Input: tee fileOne.txt
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. Then, whatever the user has entered prior to that will be written to the specified file. Since fileOne.txt already exists with some content inside, its content will be replaced with the user's input.

Description: tee with non-existing FILE
Resource Files:
Input: tee nonExistfile.txt
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. Since nonExistfile.txt file does not exist, a new file with the same name will be generated, containing the user's inputs.

Description: tee with -a Flag with FILE
Resource Files: fileOne.txt
Input: tee -a fileOne.txt
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. Then, whatever the user has entered prior to that will be appended to the end of fileOne.txt.

Description: tee with -a Flag with non-existing FILE
Resource Files:
Input: tee -a nonExistfile.txt

Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. Since nonExistfile.txt file does not exist, a new file with the same name will be generated, containing the user's inputs.

Description: tee with -a flag with multiple FILEs
Resource Files: fileOne fileTwo fileThree nonExistingFile
Input: tee -a fileOne.txt fileTwo.txt fileThree.txt nonExistingFile
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. All of the user's inputs will be appended to fileOne.txt fileTwo.txt and fileThree.txt. Since nonExistFile does not exist, a new file with the same name will be generated, containing the user's inputs.

NEGATIVE

Description: tee one valid file but with invalid option
Resource Files: fileOne.txt
Input: tee -o fileOne.txt
Expected behaviour: Output as follows
tee: Invalid flag option supplied

Description: tee one valid directory
Resource Files: myDirectory
Input: tee myDirectory
Expected behaviour: Output as follows
tee: myDirectory: Is a directory
Then continues to keep reading the user's input until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out.

POSITIVE INTEGRATION with (quoting, IO, command sub, semicolon, globbing, pipe)

Description: tee with quoting
Resource Files: fileOne.txt fileTwo.txt myDirectory
Input: tee -a "fileTwo.txt" NonExistingFile.txt 'fileOne.txt'
Expected behaviour: keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. All of the user's inputs will be appended to fileOne.txt fileTwo.txt. Since nonExistFile does not exist, a new file with the same name will be generated, containing the user's inputs.

Description: tee with IO redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: tee fileOne.txt nonExistingFile < fileThree.txt > output.txt
Expected behaviour: The content of fileThree.txt is written to fileOne.txt and nonExistingFile is created with the same content. output.txt file is created with the same content as fileThree.txt, which is as follows. (output file may not appear on intelliJ immediately)
" ' < > * | ` "
\ $ % ; , . ? /
======------

Description: tee with semicolon.
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: tee -a fileTwo.txt fileThree.txt fileOne.txt; echo hello
Expected behaviour: Tee will keep taking input from the user and when CTRL + D is hit, the entire user input is appended to the specified files. Afterwards, echo is executed.
Output as follows:
…
hello
Where … is the program displaying whatever input the user has entered for that line. Echo's result is after the user hits CTRL + D.

Description: tee with globbing
Resource Files: fileOne.txt fileTwo.txt fileThree.txt myDirectory
Input: tee -a *
Expected behaviour: Upon entering the command, the system displays a warning message for myDirectory as it is not a file. The program then keeps reading in the input of the user until CTRL + D is hit. Prior to that, whenever the user hits Enter, the newline character, the current line of input will be displayed out. All of the user's inputs will be appended to all the files in the directory, fileOne.txt, fileTwo.txt and fileThree.txt.
Output as follows.
tee: myDirectory: Is a directory
…
Where … is the program displaying whatever input the user has entered for that line. Echo's result is after the user hits CTRL + D.

Description: tee with pipe and wc
Resource Files:
Input: tee | wc
Expected behaviour: keeps taking in user's input and when CTRL + D is hit, the input is passed to wc, which will display the lines, words and bytes of all the user input.

NEGATIVE INTEGRATION

Description: tee with IO redirection but multiple files provided for input redirection
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: tee fileTwo.txt nonExistingFile < fileThree.txt fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection

Description: tee with IO redirection but multiple input redirections
Resource Files: fileOne.txt fileTwo.txt fileThree.txt
Input: tee fileTwo.txt - < fileThree.txt - < fileOne.txt > output.txt
Expected behaviour: Output as follows
shell: Too many files provided for redirection