Equivalence Partition Testing

Summary
I ran a series of tests on this project. To test that the processes are running parallel I needed to see a command entered last appear first on the output screen which it has, especially if it is an echo command while the other commands are wc or cats (processes that would generally take longer). The processes run parallel and do not occur in the same order as they are called in the command line. I did not provide the results of the tests, but they were conducted and they passed. I partitioned many inputs here.

For single commands I partitioned the arguments of echo, cat and wc into valid entries and invalid entries based on the command arguments. I did discover **one possible error** with my coding. Ending a command with ">" will state that you need a file for redirection and then prompt the user for another command. **However**, ending a command argument with the "**&**" produces a result where the "**&**" is ignored. I would argue that ignoring the "**&**" is the same as treating it as an illegal act, however if you were expecting an error message there is a problem with my code.

For single commands with redirection I assumed commands were working properly and tested to see if output was generated correctly. The partitions here are the command alongside redirection. I tested all possible combinations of the commands with creating new files (and writing to them) and writing to existing files. Based on the results from my Single Command tests the only way to generate an error by input was to end a statement with redirection.

Lastly, for parallel processing, I considered each command and redirection itself as partitions. I used a combination of testing them to satisfy (a) the requirements given to us, and (b) that all combinations of 4 variables work in tests of three or more parallel operations.

Test results are marked in the corresponding tables and the test cases themselves are stated at the bottom of the "Equivalence Partition Testing" report. Thank you for your time,

Edgar

Single Commands

	Valid	Invalid
echo	T1	T17
cat	T2	T18
wc	T3	T16

*There is no argument so large echo will not repeat it
*Since there is only one argument input all cases are covered for when there should be an input error or when the correct output should occur.

Single Commands w/ Redirection

	Valid (Existing File)	Valid (New File)
echo	T4	T7
cat	T5	T9
wc	T6	T8

Strong Eq. Partition (Valid Subdomains)

- Columns correspond to test numbers
- Numbers in columns represent the amount of commands parallel processed (& operator)

	T10	T11	T12	T13	T14	T15
echo	6	0	0	2	0	1
cat	0	6	0	2	2	1
wc	0	0	6	2	1	1
>	0	0	0	6	3	0

Test Cases

T1: echo hello
T2 cat readme.txt
T3: wc readme.txt
T4: echo>output.txt hello
T5: cat>output.txt readme.txt
T6: wc>output.txt readme.txt
T7: echo>newfile.txt hello
T8: wc>newfile2.txt readme.txt
T9: cat>newfile2.txt readme.txt
T10: echo hello&echo goodbye&echo hello&echo goodbye&echo hello&echo hello
T11: cat hello&cat goodbye&cat hello&cat goodbye&cat hello&cat hello
T12: wc readme.txt&wc readme.txt&wc readme.txt&wc readme.txt&wc readme.txt&wc readme.txt
T13: echo>output.txt hello&echo>output.txt hello&cat>output.txt readme.txt& cat>output.txt readme.txt&wc>output.txt readme.txt&wc>output.txt readme.txt
T14: cat>output.txt readme.txt&cat>output.txt readme.txt&wc>output.txtreadme.txt
T15: echo hello&cat readme.txt&wc readme.txt
T18: cat yourdog.txt
T16: wc yourdog.txt
T17: echo hello>