ERRORS CLUSTERS

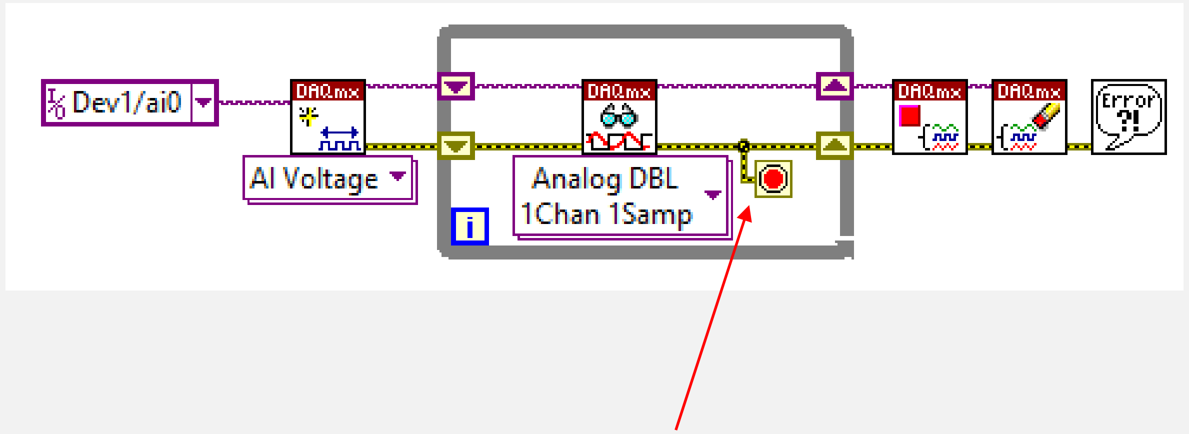
ERROR CLUSTERS

Error Clusters are typically seen with DAQmx functions. The cluster consists of 3 items . . .

1. Status: A Boolean value that reports True if an error occurred.
2. Code: A 32-bit signed integer that identifies the error numerically. If a code is thrown and Status is False, then this signals a Warning instead of an Error.
3. Source: A string that identifies where the error occurred.

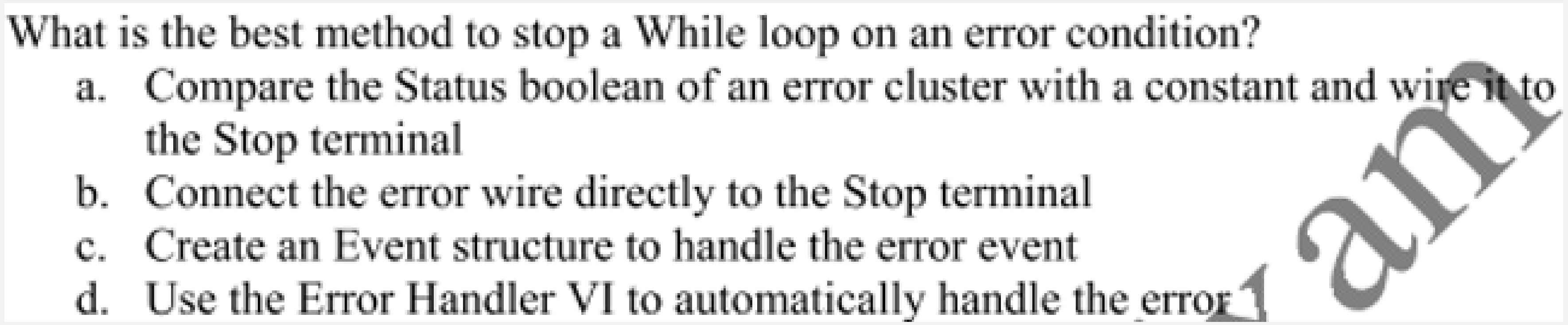
Often times, the error will propagate through all the icons until it reaches the last icon. Students typically think that the error happens at the last icon, but often times it will occur before and it's important to look at the Source.

ALWAYS PROGRAM SUCH THAT ERRORS WILL STOP YOUR PROGRAM



Notice that the error line is connected directly to the While loop conditional. You should always do this in your program. If you don't do this, then often times there will be an error (such as the DAQ board is not turned on) and your program will seem like it is working well and you may wonder why you are not getting any readings. The program actually has errors and if you would have implemented the error to the stop appropriately, then it would let you know that there was an error and stop the program.

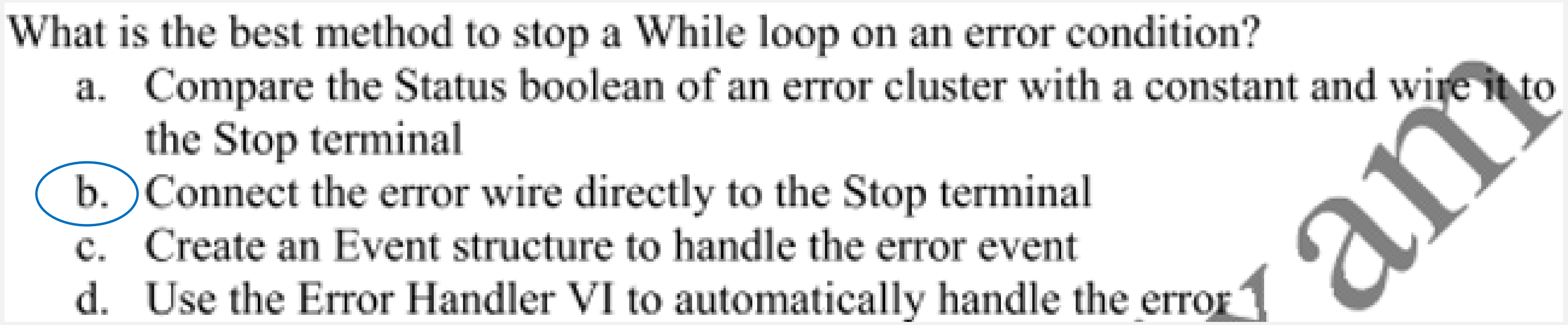
CLAD QUESTION



What is the best method to stop a While loop on an error condition?

a. Compare the Status boolean of an error cluster with a constant and wire it to the Stop terminal
b. Connect the error wire directly to the Stop terminal
c. Create an Event structure to handle the error event
d. Use the Error Handler VI to automatically handle the error

CLAD QUESTION



What is the best method to stop a While loop on an error condition?

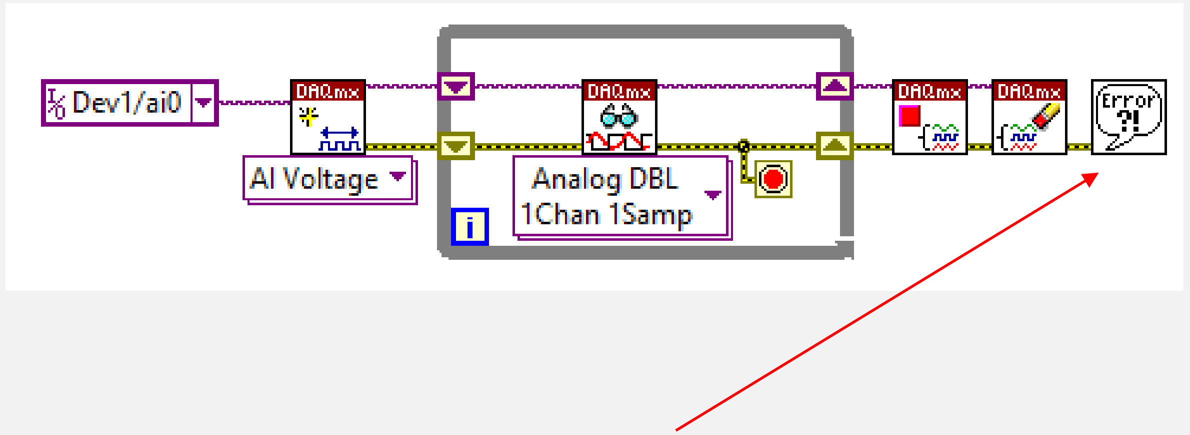
a. Compare the Status boolean of an error cluster with a constant and wire it to the Stop terminal

[✓ b.] Connect the error wire directly to the Stop terminal

c. Create an Event structure to handle the error event

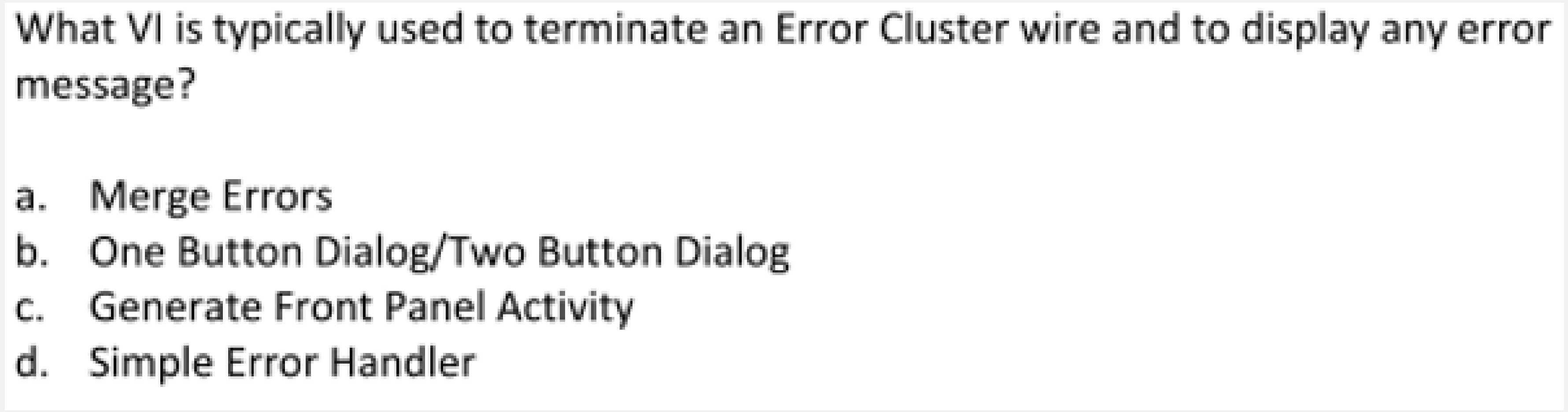
d. Use the Error Handler VI to automatically handle the error

"SIMPLE ERROR HANDLER" IS A GOOD WAY TO DISPLAY ERRORS



The "Simple Error Handler" is a good way to display errors. If there is an error, then it simply pops up and describes the error. This should be included in any program that has the error cluster.

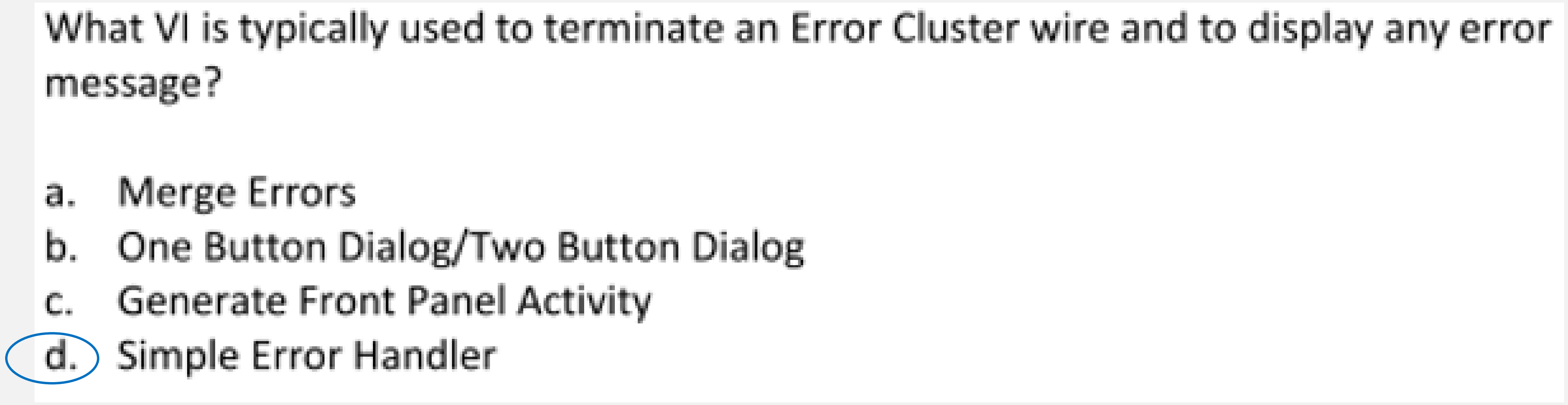
CLAD QUESTION



What VI is typically used to terminate an Error Cluster wire and to display any error message?

a. Merge Errors
b. One Button Dialog/Two Button Dialog
c. Generate Front Panel Activity
d. Simple Error Handler

CLAD QUESTION



What VI is typically used to terminate an Error Cluster wire and to display any error message?

a. Merge Errors
b. One Button Dialog/Two Button Dialog
c. Generate Front Panel Activity
[✓ d.] Simple Error Handler

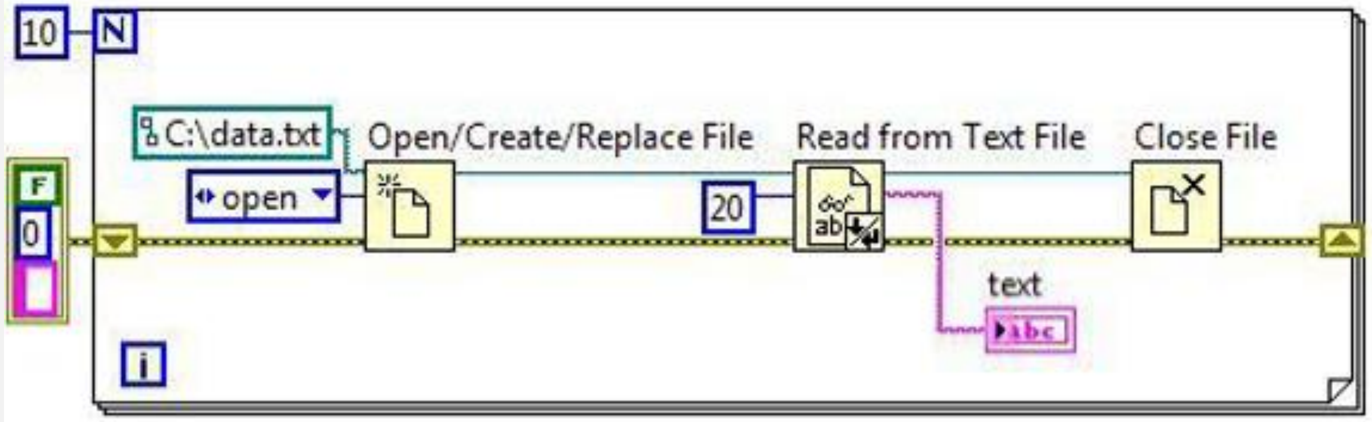
AUTOMATIC ERROR HANDLING

LabVIEW defaults programs to automatic error handling. Automatic Error Handling makes programs such that if there is an error, then it will suspend execution and display an error box (it will also highlight the last VI that the error passed to, but the error could have originated in a previous VI so be sure to read what the source of the error is).

Although Automatic Error Handling will display the error, it is still better practice to use the Simple Error Handler VI in all of your programs.

CLAD QUESTION

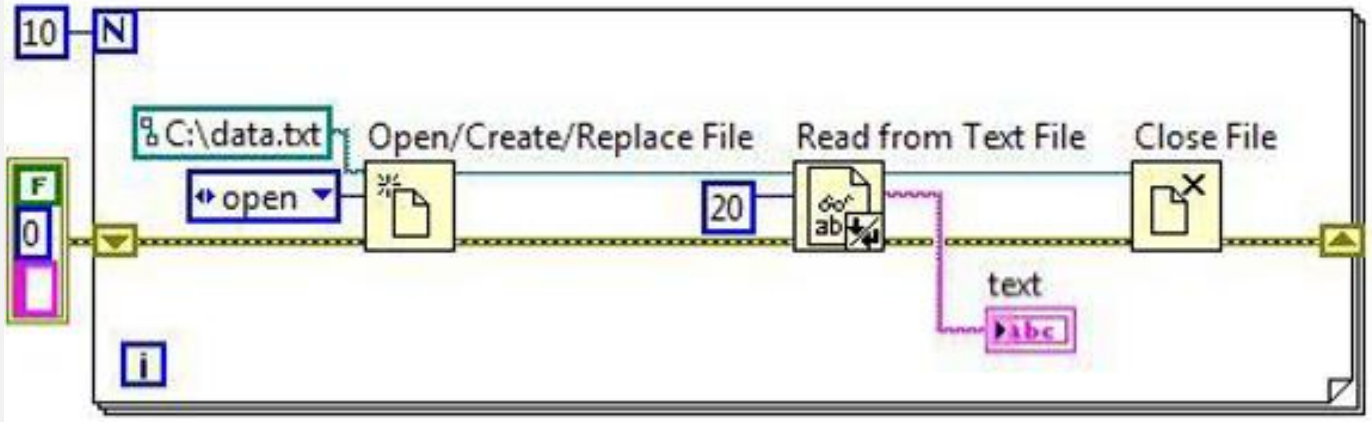
For the VI shown in the following block diagram, automatic error handling is enabled. If the file C:\data.txt does not exist, will an error dialog box pop up?



a. Yes, each time the error from the Open/Create/Replace File function is returned
b. Yes, each time a loop iteration completes
c. Yes, but only once, not on each iteration of the loop
d. No

CLAD QUESTION

For the VI shown in the following block diagram, automatic error handling is enabled. If the file C:\data.txt does not exist, will an error dialog box pop up?

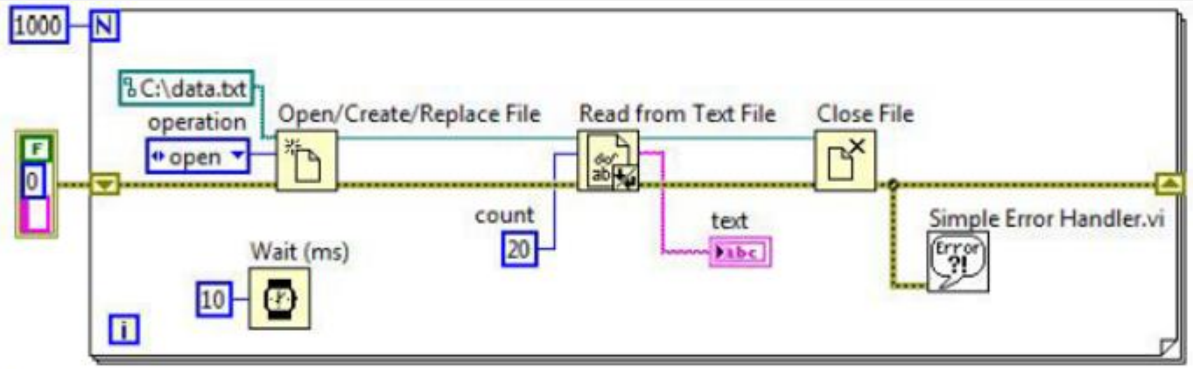


Needs to include the Simple Error Handler Function

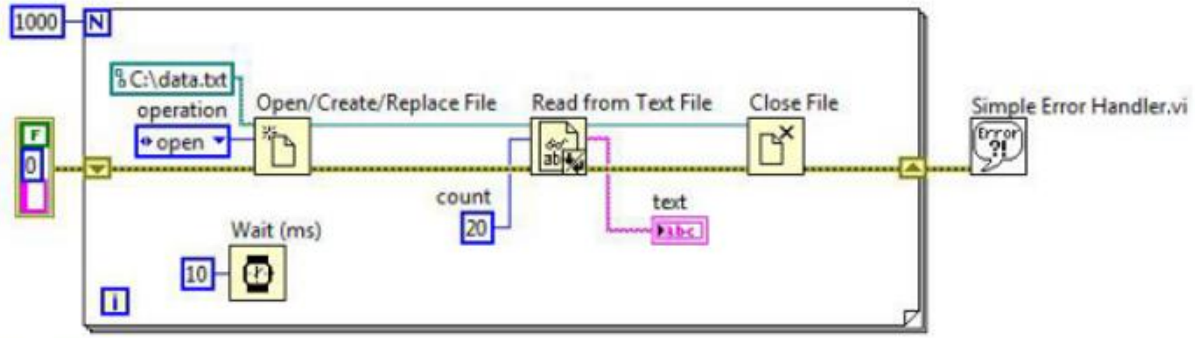
a. Yes, each time the error from the Open/Create/Replace File function is returned
b. Yes, each time a loop iteration completes
c. Yes, but only once, not on each iteration of the loop
[✓ d.] No

CLAD QUESTION

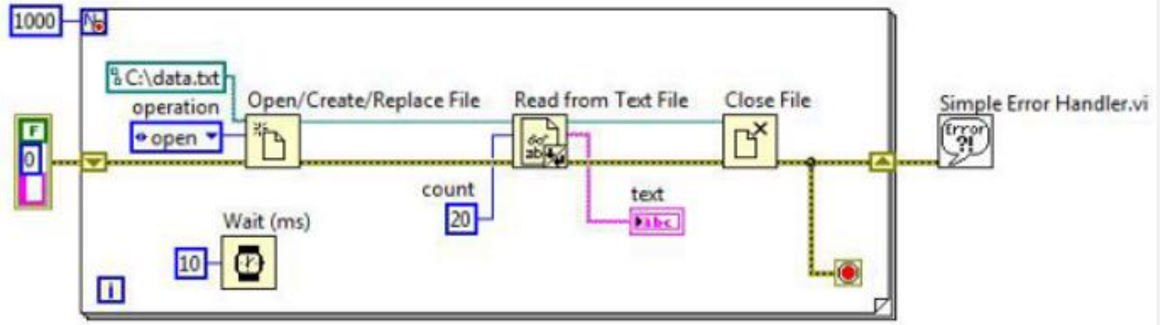
The file C:\data.txt does not exist, but the VI does not report an error. Which code snippet reports an error and stops?



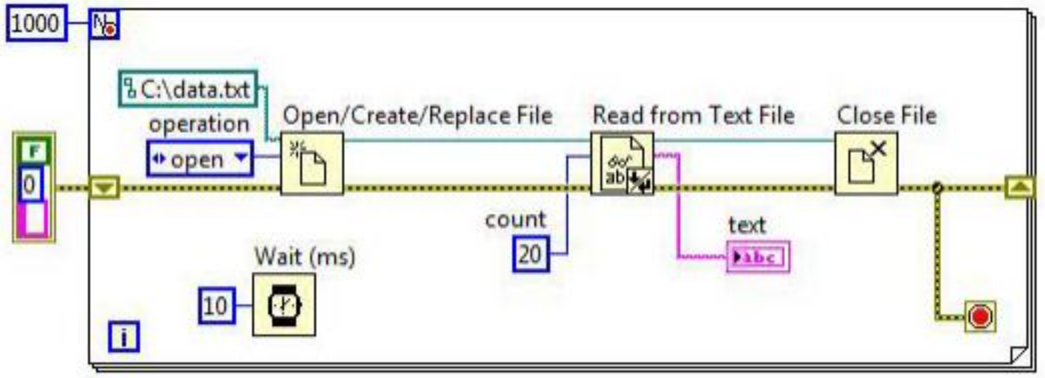
a.



b.



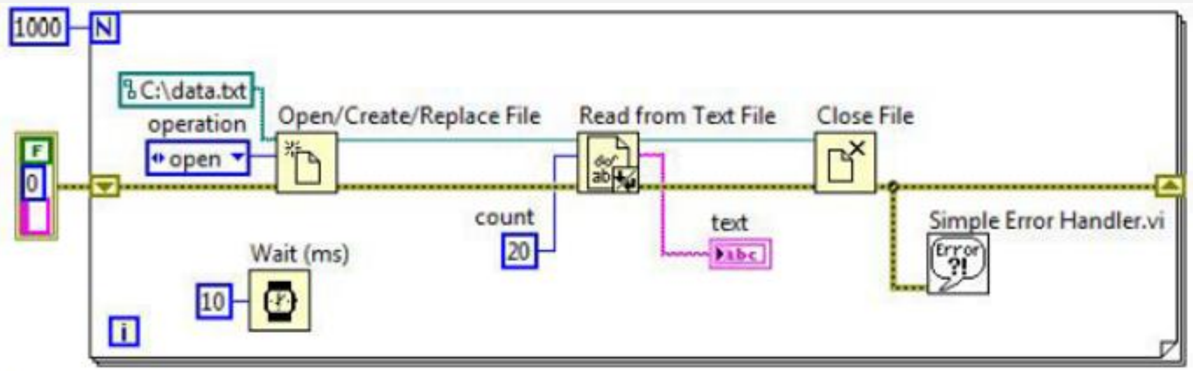
c.



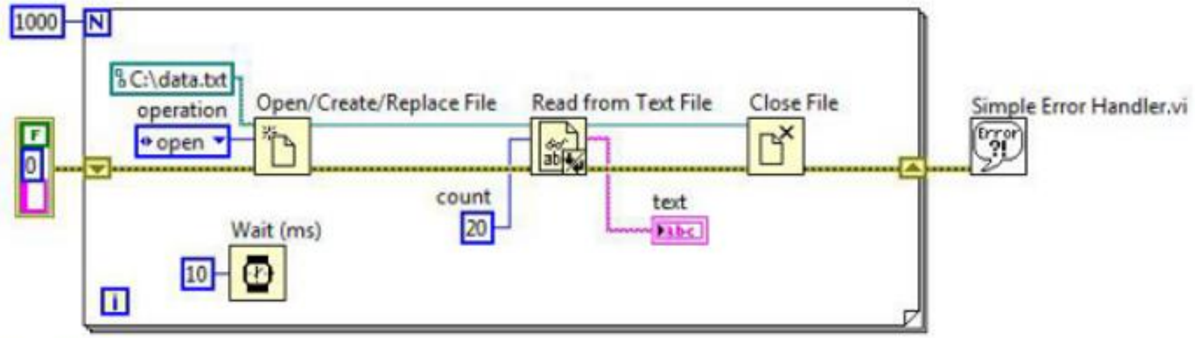
d.

CLAD QUESTION

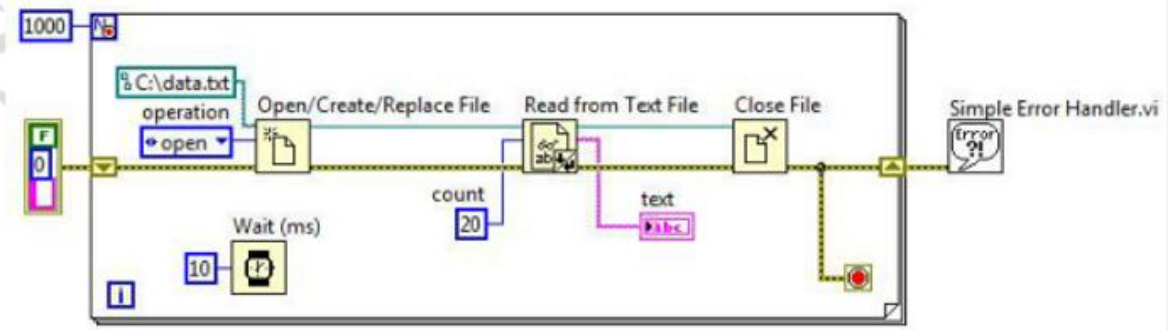
The file C:\data.txt does not exist, but the VI does not report an error. Which code snippet reports an error and stops?



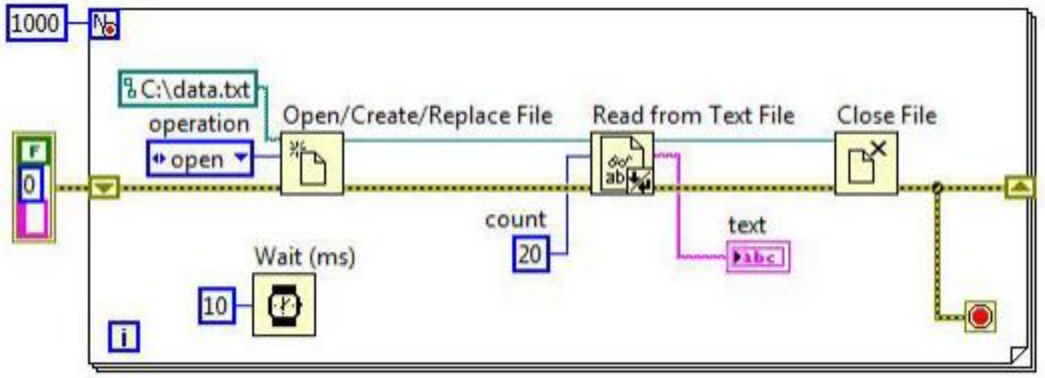
a.



b.



[✓ c.]

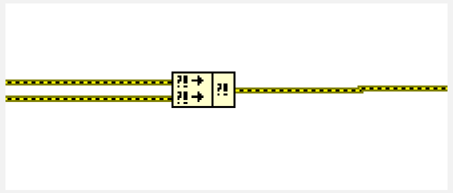


d.

MERGE ERROR VI

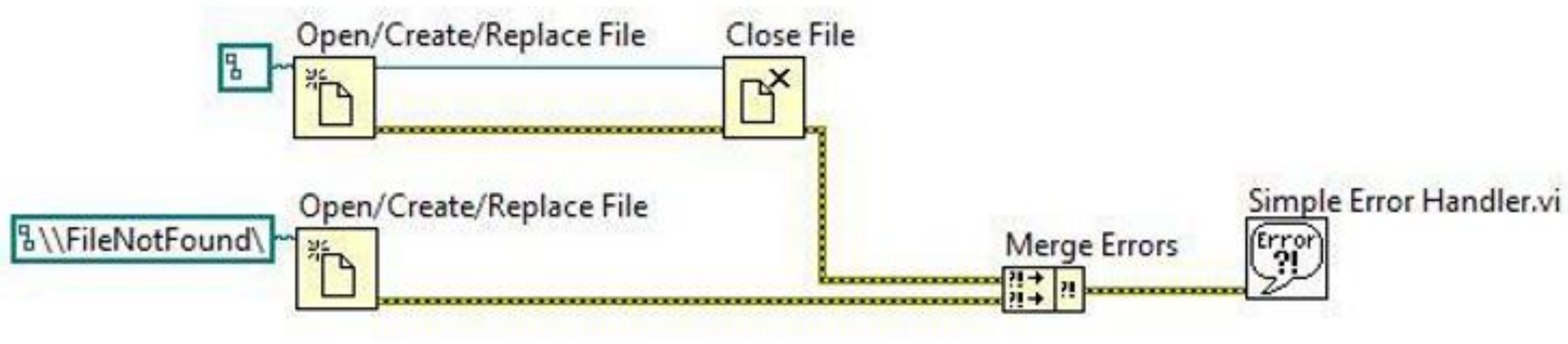
Sometimes it’s beneficial to have 2 or more errors merge together. This can be achieved with the Merge Error VI. If more than one line has an error, then the output of the Merge Error VI will output the topmost error.

You can drag this box down and merge more than 2 errors.



CLAD QUESTION

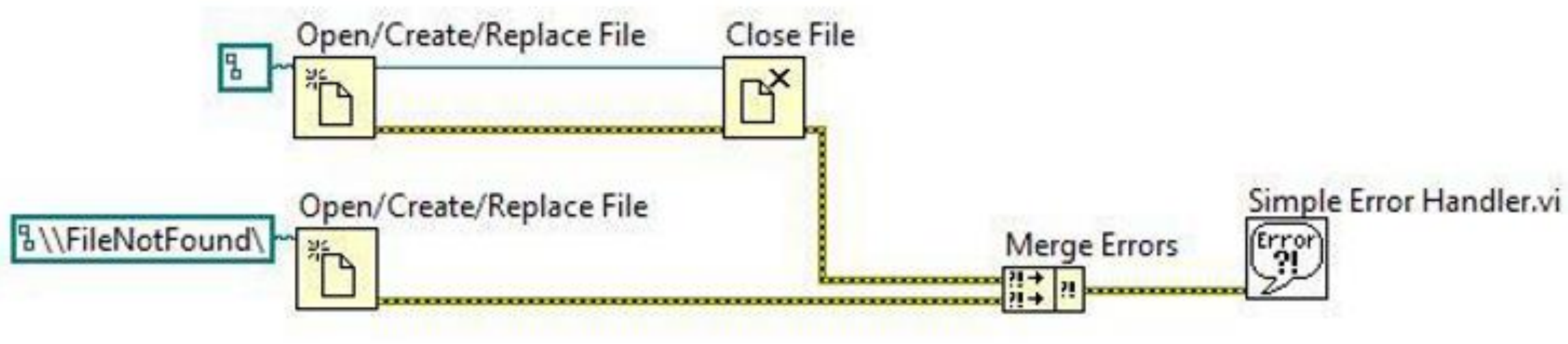
Q4: How many errors does LabVIEW display at the end of execution?



A No Errors
B One Error
C Two Errors
D Three Errors

CLAD QUESTION

Q4: How many errors does LabVIEW display at the end of execution?



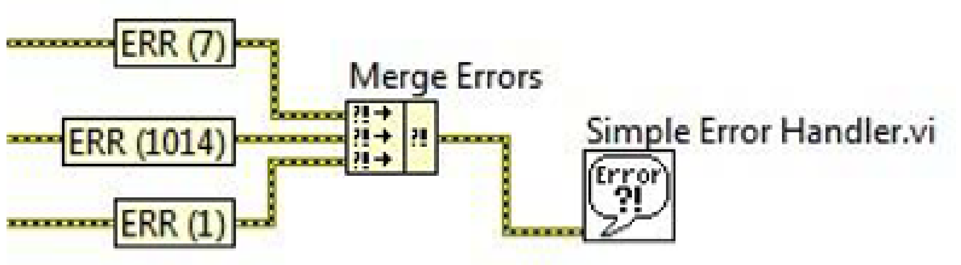
[✓ **B**] One Error

A	No Errors
[✓ **B**]	One Error
C	Two Errors
D	Three Errors

CLAD QUESTION

Q14: You have written a LabVIEW VI with three parallel code paths. All three error cluster wires are wired into a Merge Errors function.

What error is reported to the user?



A Error 1014, because Merge Errors outputs the first error to occur chronologically

B Error 7, because Merge Errors outputs the first error wired in to it from the top down

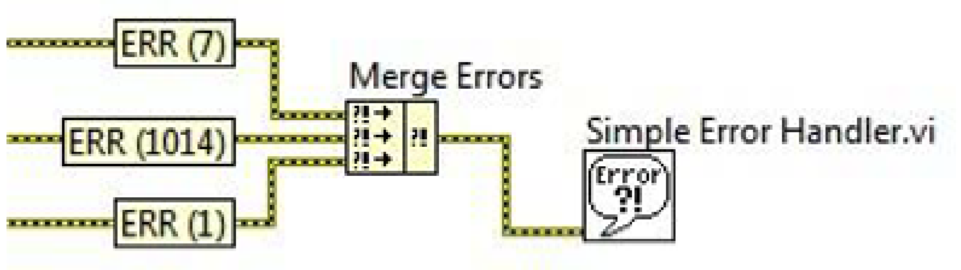
C Error 1, because Merge Errors outputs the most significant error of the errors wired in to it

D All three errors are reported, because Merge Errors concatenates the errors into a single error message to display to the user

CLAD QUESTION

Q14: You have written a LabVIEW VI with three parallel code paths. All three error cluster wires are wired into a Merge Errors function.

What error is reported to the user?



A Error 1014, because Merge Errors outputs the first error to occur chronologically

[✓ **B**] Error 7, because Merge Errors outputs the first error wired in to it from the top down

C Error 1, because Merge Errors outputs the most significant error of the errors wired in to it

D All three errors are reported, because Merge Errors concatenates the errors into a single error message to display to the user