SFT 221

Workshop 6

Name:	patel vrundaben vijaykumar
ID:	158605220
Email:	vvpatel20@myseneca.ca
Section:	Ngg

Authenticity Declaration:
I declare this submission is the result of my own work and has not been shared with any other student or 3rd party content provider. This submitted piece of work is entirely of my own creation.

Updated

Line Number	Filename	Updated Line
8	main.c	char testStr[] = "This is a\n string with embedded newline characters and\n12345 numbers inside it as well 7890";
14	main.c	printUntil(index.str, index.lineStarts[i], '\0');
19	main.c	l4cInfo(&log, "LINES");
12	main.c	for (i = 0; i < index.numLines&&i<MAX_INDEX_SIZE; i++)
18	main.c	for (i = 0; i < index.numWords && i < MAX_INDEX_SIZE; i++)
16	main.c	if (l4cCheck(&log, errMsg)) printf("%s\n", errMsg);
30	main.c	assert(strcmp(printedOutput, testStr) == 0);
58	main.c	}assert(matchFoundW);

l code is below

What Was Wrong	VS Techniques Used
In the line char testStr[] = { "This is a\n ..."};, the array testStr is being initialized with a list of characters enclosed in curly braces, which results in a syntax error. he bug occurred because the string initialization was enclosed in curly braces, which is not the correct way to initialize a string in C. The corrected code initializes testStr as a string literal.	inspection
The printUntil function was using '\n' as the line termination character, which didn't match the null-terminated string. The line termination character was corrected from '\n' to '\0' to match the null-terminated string.	breakpoints
The code lacked detailed debugging information, making it difficult to identify issues. Log4c was introduced to add informative log messages for debugging purposes. The "LINES" section was updated with a log statement.	log4c file
The original code lacks a check to ensure i doesn't exceed the MAX_INDEX_SIZE limit, leading to potential buffer overflow. i fixed it by adding a condition to prevent exceeding the limit.	breakpoints
The original code did not prevent i from exceeding the MAX_INDEX_SIZE limit, leading to potential buffer overflow. I fixed it by adding a condition to avoid exceeding the limit.	breakpoints
t there was no mechanism to log error messages. This made it challenging to keep track of errors. To fix this, I introduced conditional logging. It uses l4cCheck to check if there are any error messages stored in errMsg. If there are, it logs the error messages using Log4c, ensuring they are recorded for further analysis.	log4c file
he assert statement checks if printedOutput matches testStr.	assertion
The assert statement checks if matchFoundW is true.	assertion

Did you find more bugs using the log file or assertions? Why did one work better than the other?

I found bugs equally using the log files and assertions. Assertions immediately stop the program when something goes wrong, making it easier to spot and fix problems early. It's like having a red flag that signals trouble. Log files, on the other hand, are useful in live systems to track issues when the program is running. They are like a record of what happened, but they don't immediately point out the problem. So, assertions are great for finding and fixing bugs while writing the code, while log files are better for keeping an eye on things when the program is up and running.

Do you like to debug using log files or the debugger? Why do you prefer one over the other? Can you envision a situation where you would only be able to use log files? Would you be able to debug as well? Would you be able to debug at the same speed as using a debugger, slower or faster?

I prefer using debuggers because they offer real-time, interactive tools for identifying and fixing issues quickly. Debuggers allow me to set breakpoints, inspect variables, and step through code, which speeds up the debugging process. While log files are helpful, they often involve a slower process. Analyzing logs, making code adjustments, and repeating the cycle takes more time. Debuggers provide immediate control and visibility into code behavior during development, making it faster and more efficient. If limited to log files, debugging could become slower due to the lack of real-time insights

How did you gain confidence that you had found all the bugs? Would you bet your life savings that you found all the bugs? If not, what would you do so that you would feel comfortable betting that you had found all the bugs?

I can't be completely sure I've found all the bugs because software can be tricky, and new issues can pop up. To feel more confident, I'd do a few things. First, I'd have lots of different people test the software because they might spot things I missed. Second, I'd ask my colleagues to check my code for errors. Third, I might use special tools to help me find common problems. Finally, I'd pay attention to feedback from users and keep an eye on how the software behaves in the real world. These steps would make me more sure, but I can't be 100% certain because software is always changing.