SFT 221

Workshop 5

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 code is below

Line Number	Filename	Updated Line
48	stringhelp.c	if (isNumber(str + i, sp - i))
47	stringhelp.c	sp = nextWhite(str + i) - 1;
21	stringhelp.c	for (i = 0; i < len; i++) {
80	stringhelp.c	for (i = 0; i < sp-start; i++)
84	stringhelp.c	word[i] = '\0';
95	stringhelp.c	for (i = 0; i < sp - start; i++) {
98	stringhelp.c	word[i] = '\0';

What Was Wrong	VS Techniques Used
in the original code, the calculation of the number's length was incorrect because it included an extra '+1', leading to incorrect results. To fix this i adjusted the length calculation to sp - i, ensuring the correct length of the number	i used breakpoints
The original code calculated sp incorrectly, not considering the space character. To fix this, I subtracted 1 from sp to ensure it points to the correct position before the whitespace.	i used step through
The original code had an incorrect loop condition, using result which was unnecessary. I fixed this by removing result from the loop condition, ensuring the loop iterates over the correct range.	i used breakpoints
The original code had issues copying words correctly because it didn't account for the starting position. To fix this, I adjusted the loop and copying operation to correctly extract words.	i used breakpoints
The original code had incorrect null-termination for words. It null-terminated at the wrong position. I corrected this by null-terminating the word at the correct position in the array.	i used step through
The original code had issues copying numbers correctly because it didn't account for the starting position. To fix this, I adjusted the loop and copying operation to correctly extract numbers	i used breakpoints
The original code had incorrect null-termination for numbers. It null-terminated at the wrong position. I corrected this by null-terminating the number at the correct position in the array.	i used step through

What was the most useful debugger tool you used to find the bugs? Why was it more useful than other techniques you tried?

the most useful debugger tool was setting Breakpoints in the code. Breakpoints let me stop the program at specific lines, allowing me to check the values of variables and see how the code behaved step by step. This helped me identify where the code was going wrong and why. Breakpoints were more useful because they gave me direct control and insights into the code's execution, making it easier to pinpoint and fix the issues. Other techniques, like just reading the code or using print statements, don't offer the same level of control and visibility, so breakpoints were the most effective way to find and fix the bugs.

Which debugger features did you NOT use? Why did you not use these features? For each feature you did not use, give an example of a problem you would use it for.

I did not use Watch Variables and Conditional Breakpoints during debugging. I didn't use Watch Variables because I could check variable values using regular breakpoints, and it wasn't necessary to have a separate list for watching specific variables. I also didn't use Conditional Breakpoints because the bugs I encountered were straightforward and didn't require breaking based on a condition. Conditional breakpoints are handy when you want to stop the program only when a particular condition is met, like when a variable reaches a specific value. If I had a complex bug related to specific variable states, I would have used conditional breakpoints to pause the program when that condition was satisfied for debugging purposes.

Which do you think is a faster way to find bugs:
a. Using the debugger alone,
b. Using print statements alone,
c. Using the debugger and print statements?
Justify your answer.

Using the debugger alone is typically a faster way to find bugs compared to using print statements alone or using both the debugger and print statements. The debugger allows me to examine the code step by step, inspect variables, and identify issues directly within the code execution. It provides a more efficient and precise way to locate and understand the bugs. Print statements, while helpful, require me to manually insert and remove them, and they can clutter the code. Combining both methods can be useful in some cases, but it can also add complexity and may not necessarily be faster than using the debugger alone. In most cases, relying on the debugger alone is quicker and more effective for bug detection and resolution.

How did you test the program to find the bugs and to make sure you had fixed the bugs? Did you use any additional test data other than that supplied? If so, describe the techniques you used to create the test data. How confident are you that you found all the bugs?

To test the program and find the bugs, I used the provided test data and relied on the debugger extensively. I set breakpoints at critical points in the code to examine variable values and step through the code's execution. This helped me identify and fix the known bugs. However, I did not use any additional test data beyond what was supplied, and I'm not confident that I found all possible bugs.