After examining the solution to assignment two made by Perplexica, I've learned several important lessons about improving lexer design and implementation. The most significant change is the shift towards a more modular and maintainable structure. By introducing an enumeration for token types and an array of token names, I've seen how to make the code more readable and easier to debug.

I've realized the importance of expanding functionality in a lexer. The addition of more keywords and improved number parsing demonstrates how to make the lexer more comprehensive. The inclusion of string literal support shows me how to handle more complex language constructs.

Error handling is another crucial aspect I've learned about. The addition of basic error reporting for unrecognized characters is an effective way to improve the robustness of the lexer. This feature would be very helpful during language development and debugging.

The enhanced main function has taught me about better token handling and memory management. I now understand the importance of properly freeing allocated memory, especially for string tokens, to prevent memory leaks.

I've also learned about the value of file input support. By allowing the lexer to read from a file specified as a command-line argument, I can see how to make the tool more flexible and useful in real-world scenarios.

The addition of whitespace handling has shown me how to ignore certain characters that aren't relevant to the language syntax but are necessary for human readability.

Perhaps most importantly, I've gained insight into the process of iterative improvement in lexer design. This updated code demonstrates how to take a basic lexer and enhance it step by step, adding features and improving its functionality without completely rewriting it.

Overall, this exercise has given me a deeper appreciation for the complexities involved in lexer design and the importance of creating flexible, maintainable, and feature-rich tools for language processing.