A screen shot of a computer

AI-generated content may be incorrect.Screenshot of code before:

Screenshot of after:  
A screenshot of a computer

AI-generated content may be incorrect.

For this assignment, I conducted a static code analysis on the provided QuestionableCode.cpp file using both Visual Studio’s built-in analysis and CppCheck 2.16. The goal was to identify coding errors, compare the results from both tools, and apply fixes to improve code quality and security. Visual Studio primarily caught syntax-related errors and logical mistakes but missed deeper issues like vector iterator invalidation and unsafe memory operations. Conversely, CppCheck identified additional issues, including recursion risks, uninitialized pointers, memory leaks, and logic flaws, making it particularly useful for detecting hidden vulnerabilities.

Several issues were present in the original code, such as endless recursion, out-of-bounds array access, returning a pointer to a local variable, invalid iterator usage, and unsafe memory operations. I resolved these problems by modifying recursive calls, adding boundary checks, correcting iterator usage, initializing pointers, and aligning exception behavior with function declarations. For example, vector\_test() was modified to update iterators correctly after erasing an element, and work\_with\_arrays() was updated to prevent buffer overflow. Additionally, the noexcept specifier was removed from MySpecialType::DontThrow() to avoid undefined behavior when throwing an exception.

By implementing these fixes, the code is improved…. but still needs work. . This analysis demonstrated the importance of automated static code analysis in catching errors that might otherwise go unnoticed. Using both Visual Studio and CppCheck together provided a more comprehensive review of the code, ensuring that common mistakes were caught early in the development process.