**LAB TERMINAL**



**Submitted to: Sir Bilal Haider**

**Submitted by: Sara Ali**

**Reg: Fa21-bcs-011**

**Course: CC**

**Dated: 03-01-2025**

**Ans3)**

**Optimizations in Compiler:**

**1-Constant folding:**

Constant folding simplifies expressions with constant values at compile time. Instead of calculating the value at runtime, the compiler evaluates it during compilation and replaces the expression with the result.

**Before optimization:**

int main() {

int a = 3 \* (2 + 1); // This expression can be computed at compile time.

return a;

}

**After optimization:**

int main() {

int a = 9; // Replace 3 \* (2 + 1) with its result.

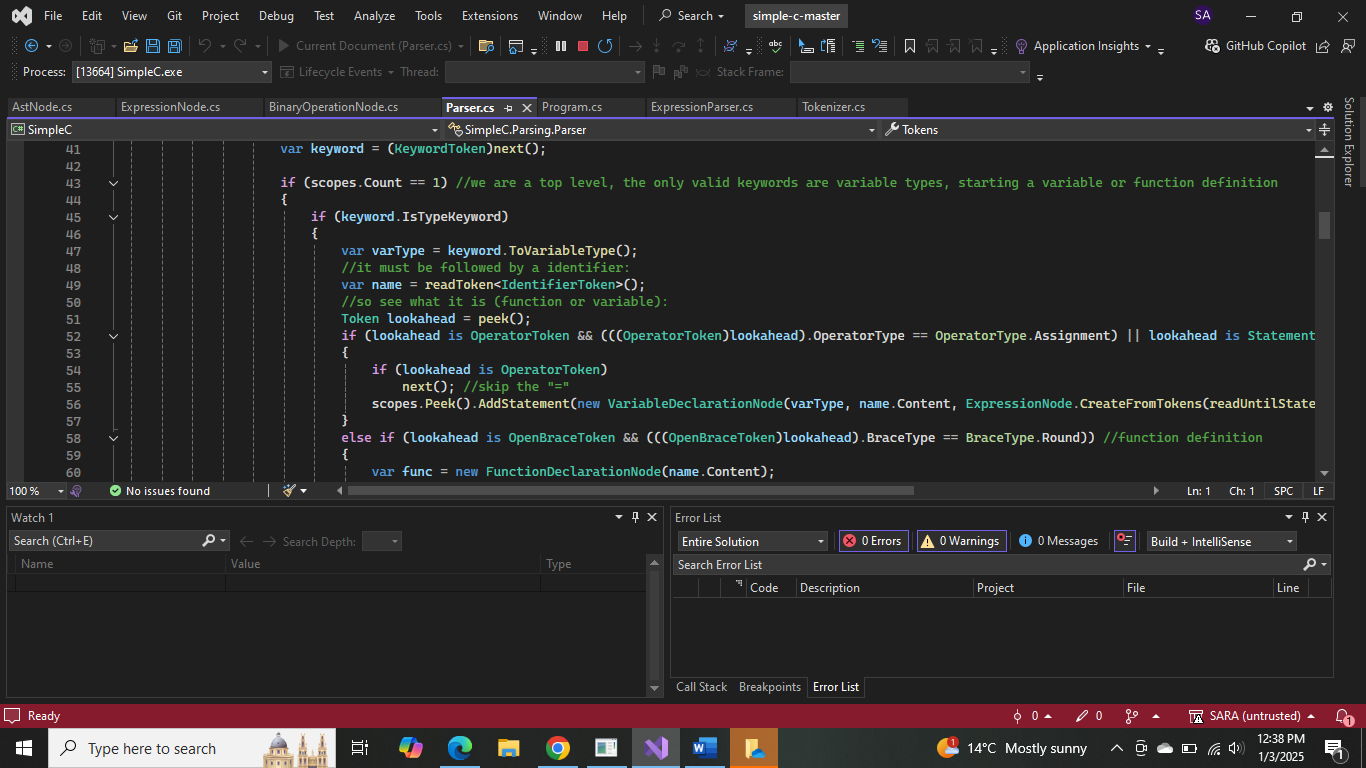
return a;

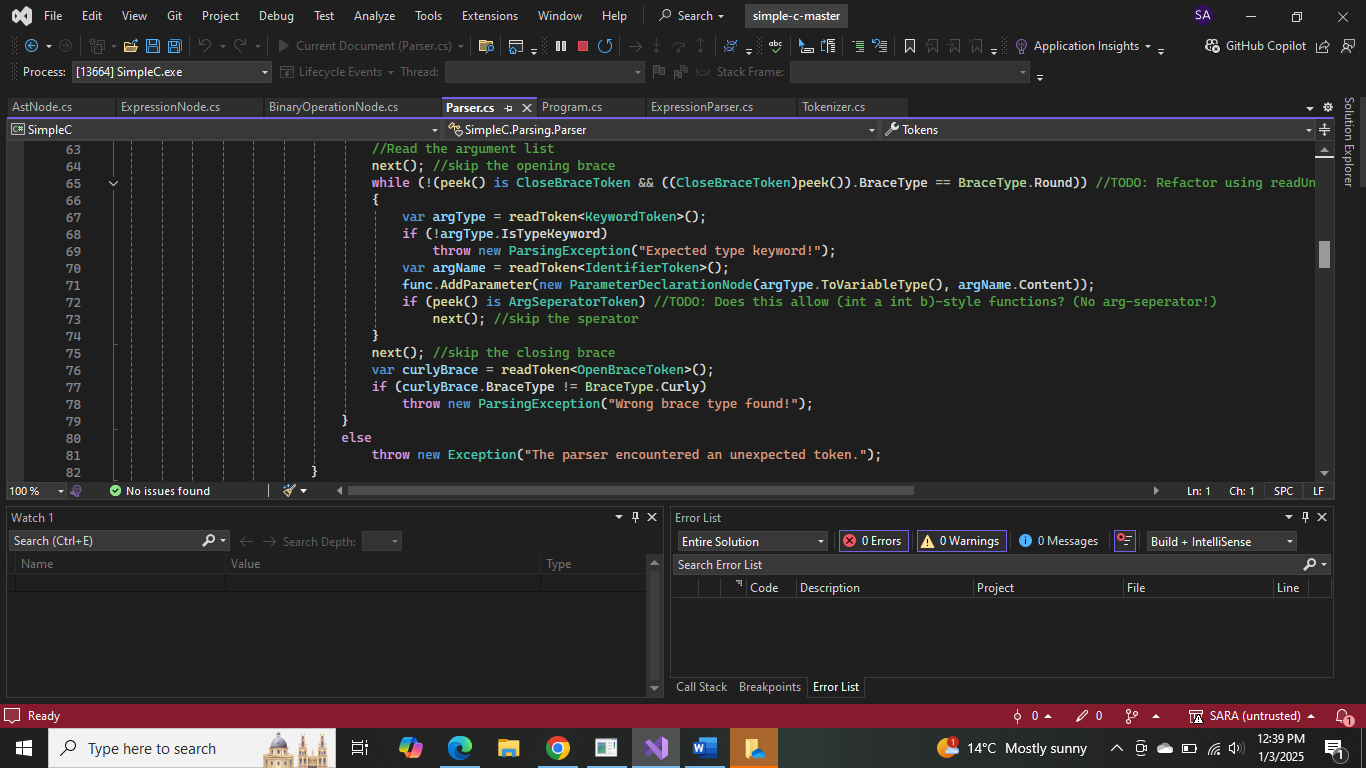
}

**Implementation in the Compiler**: During the parsing phase, detect constant expressions in the AST and compute their values.

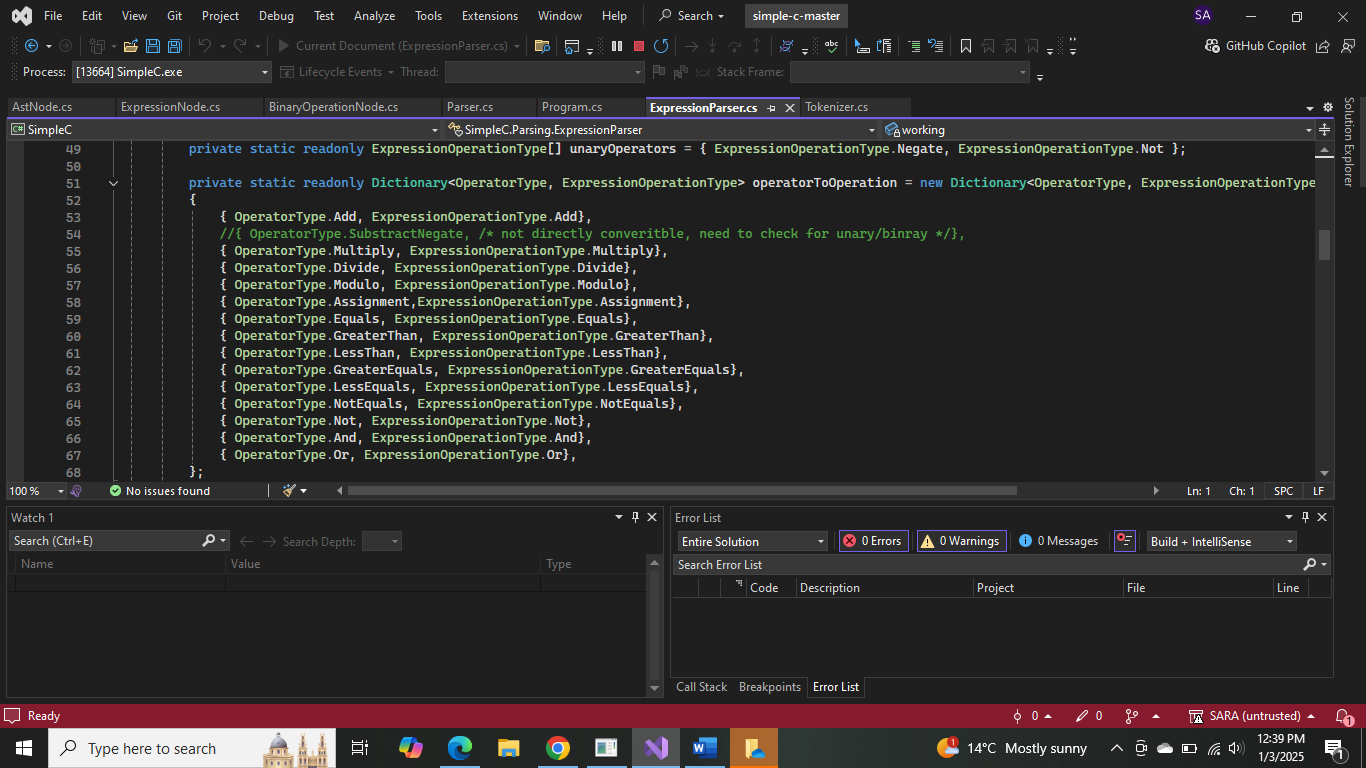
Optimization is done in:

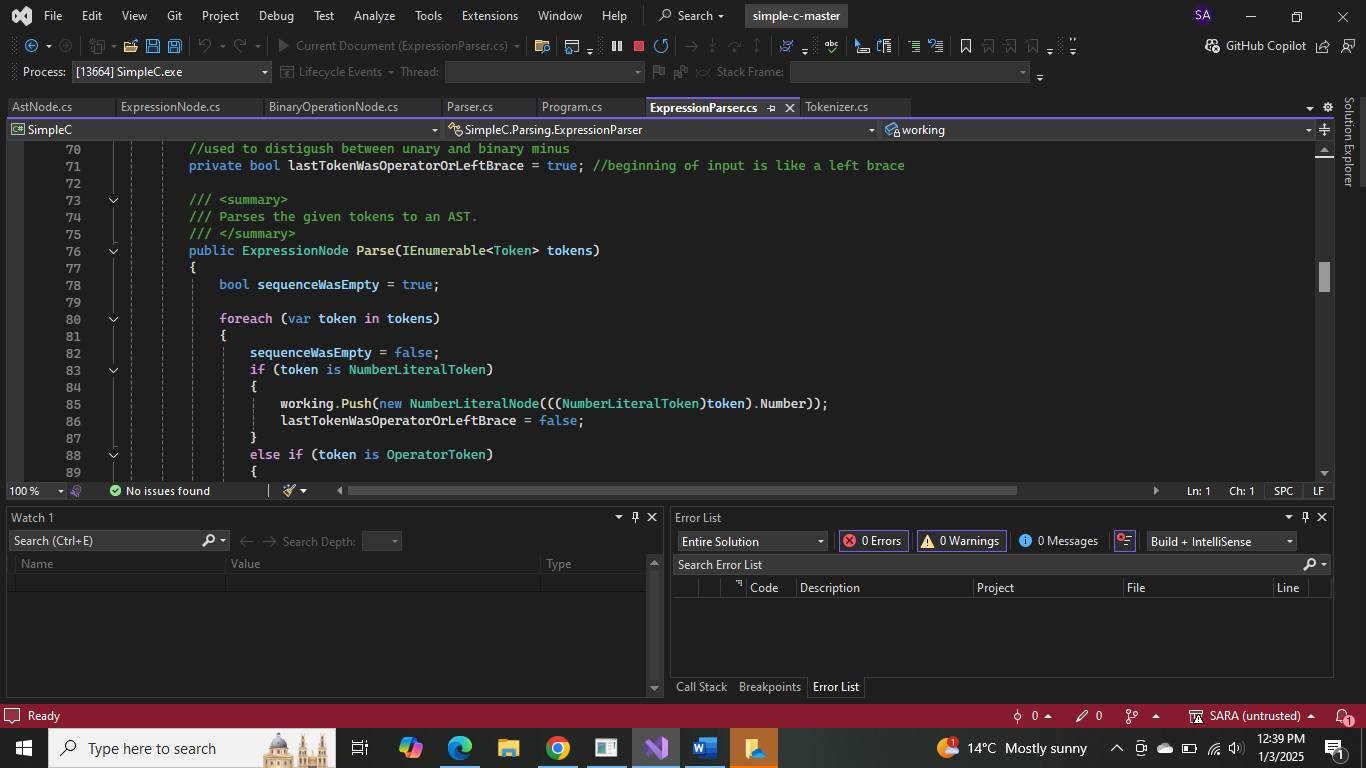
* Parser.cs: All methods or functions that traverse binary expressions BinaryExpressionNode and simplify them by evaluating constant operands.





**ExpressionParser.cs**





**2)Dead Code Elimination:**

Dead code elimination removes parts of the program that are never executed or have no effect on the output. This reduces the code size and improves runtime performance.

Example:

**Implementation in the Compiler:** Analyze the control flow and remove unreachable or unnecessary code during the code generation phase.

if (conditionNode is ConstantNode constant && constant.Value == false)

{

return; // Remove the code block associated with the condition.

}