Three Address Code Generator

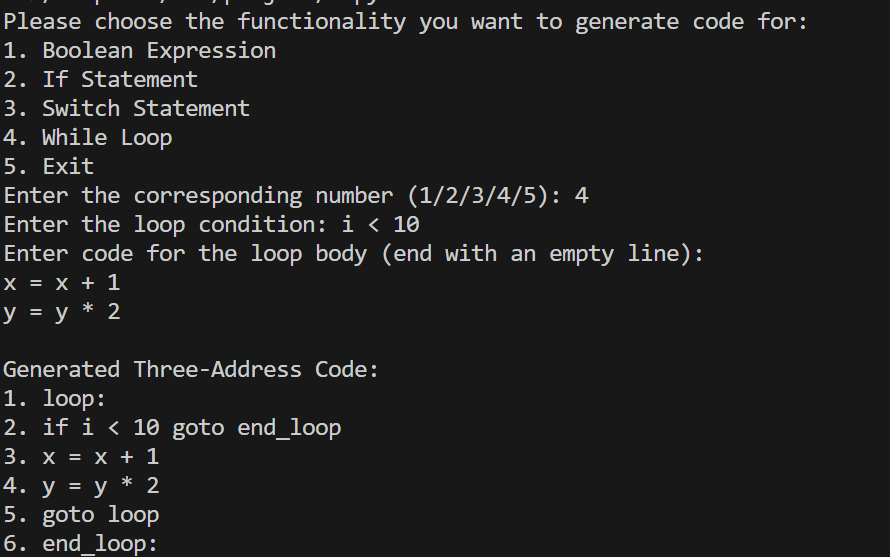
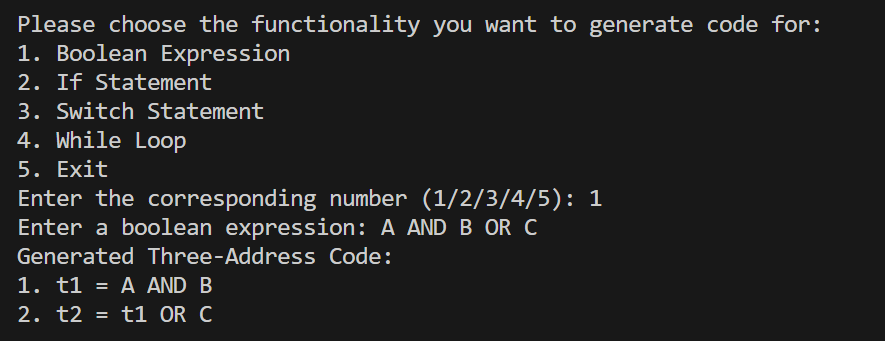
--Sahiti Lakkoju (21WU0101028)

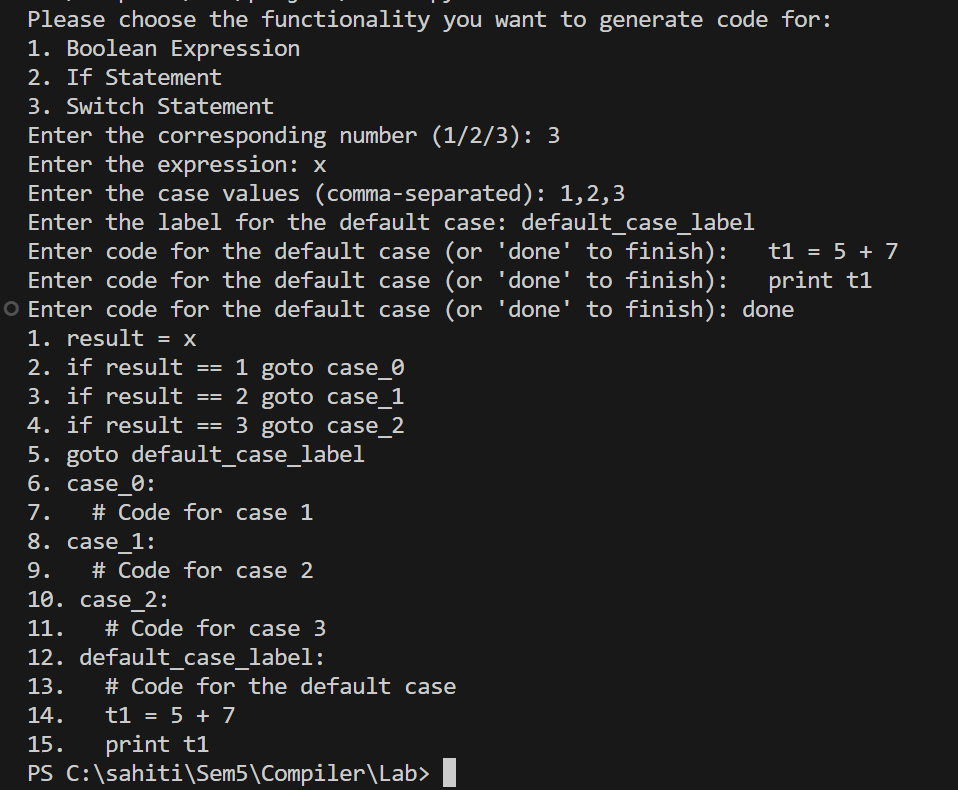
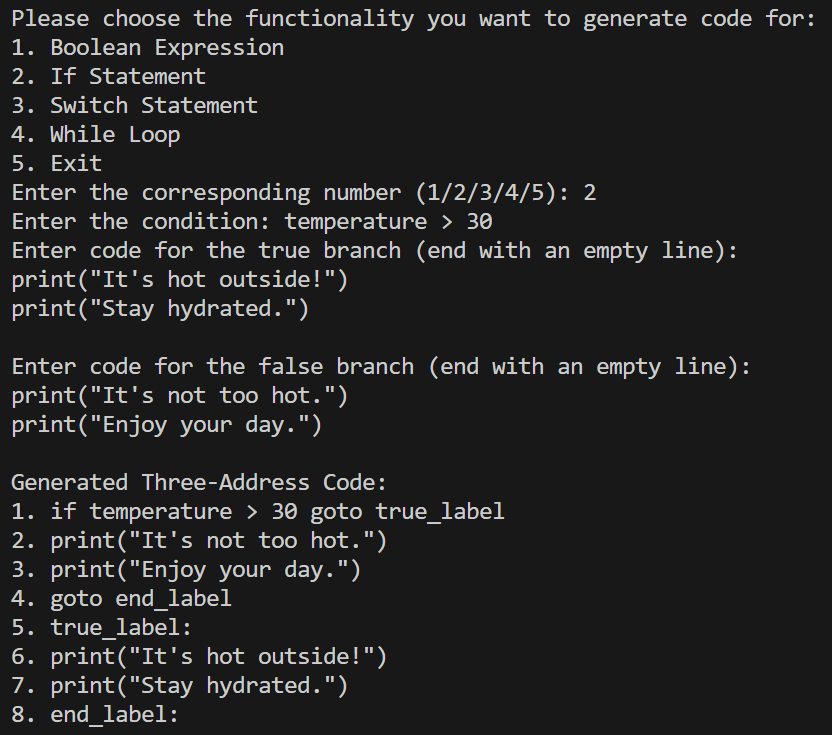
1. **Problem Statement:**

To create three address code generators for Boolean expressions, control flow constructs (e.g., if-else statements and loops), and case statements. The primary goal is to simplify the generated code, reduce redundancy, and optimize execution efficiency. This problem is pivotal in enhancing the performance and maintainability of compiler-generated code.

1. **Applications:**

Three address code generators have widespread applications in compiler design:

1. Optimization: TAC serves as an intermediate representation during optimization phases. It allows the compiler to analyze code and perform optimizations to enhance the generated code's performance.
2. Code Generation: TAC is used as an intermediate representation during code generation. It helps the compiler produce target-specific code that is both correct and efficient.
3. Debugging: TAC aids in debugging by providing a more readable and understandable low-level representation of code. Developers can trace program execution and identify errors or issues.
4. Language Translation: TAC can be used to translate code between programming languages. By translating code to a common intermediate form, it simplifies the process of translating code to multiple target languages.
5. **Results:**



* We successfully generated three address code for a range of Boolean expressions, control flow constructs, while statement and case statements.
* Performance analysis indicates a reduction in code size and improved execution speed.

1. **Future Scope:**
2. Enhancements: Planned to explore further optimizations and code generation techniques.
3. Language Support: Expanding the generators to support a broader range of programming languages.
4. Integration: Generators can be integrated into popular compiler frameworks for practical use.