****

**University Of Asia Pacific**

**Department of Computer Science and Engineering**

**Course Code:** CSE 430

**Course Title:** Compiler Design Lab

**Lab Exercise:** Compiler Design Lab Project

**Submitted To,**

Jayonto Dutta Plabon

Lecturer, Department of CSE

University of Asia Pacific

**Submitted By,**

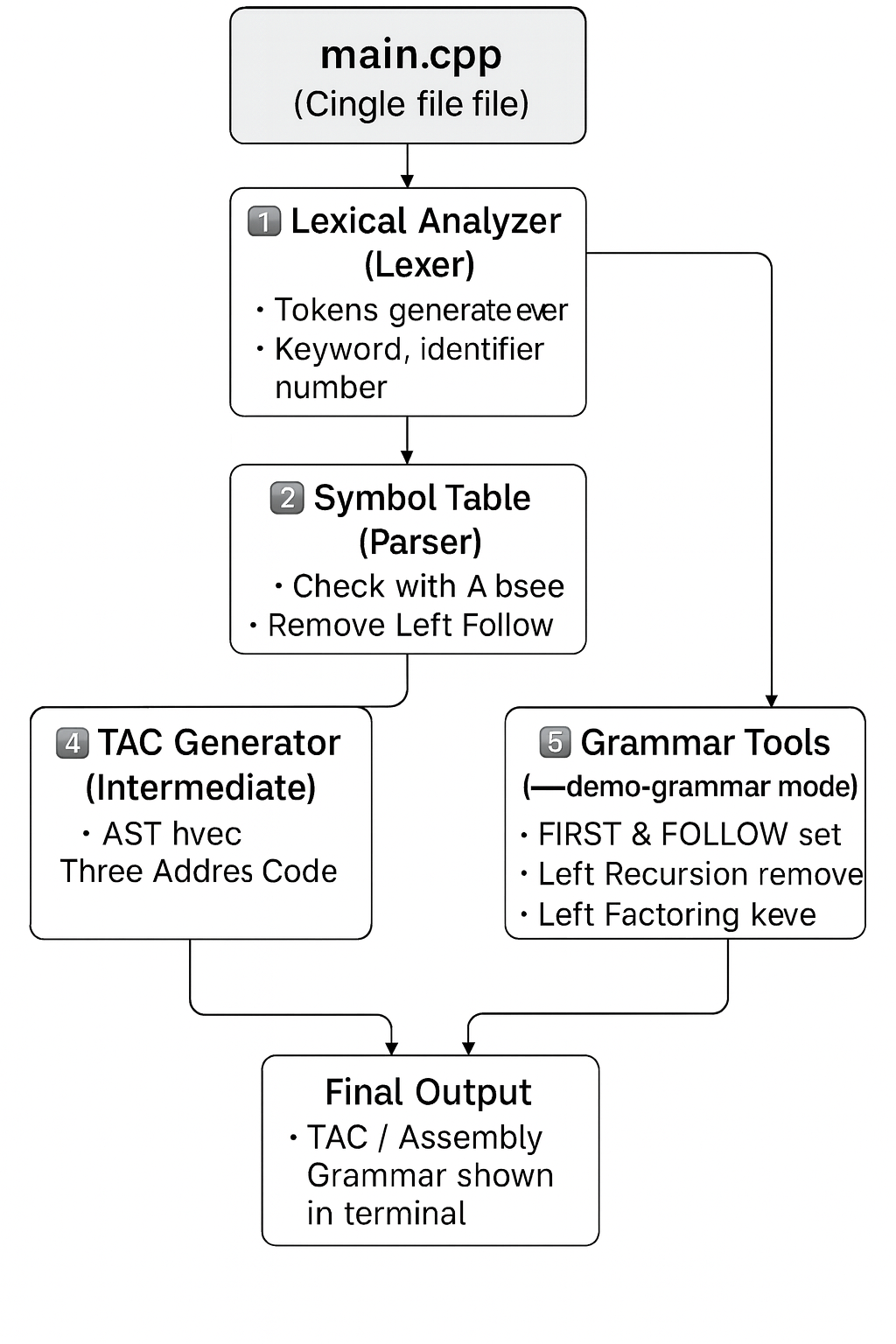
**Name:** Sadia Islam Priya

**ID:** 21201181

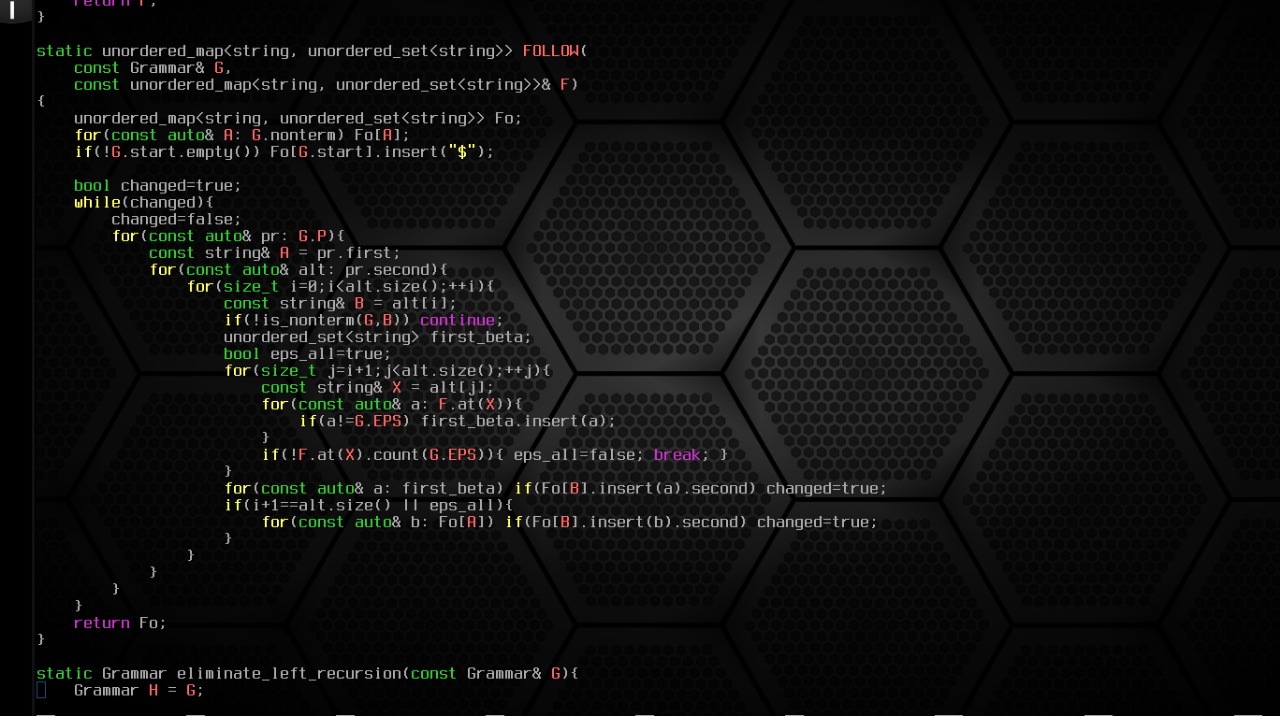
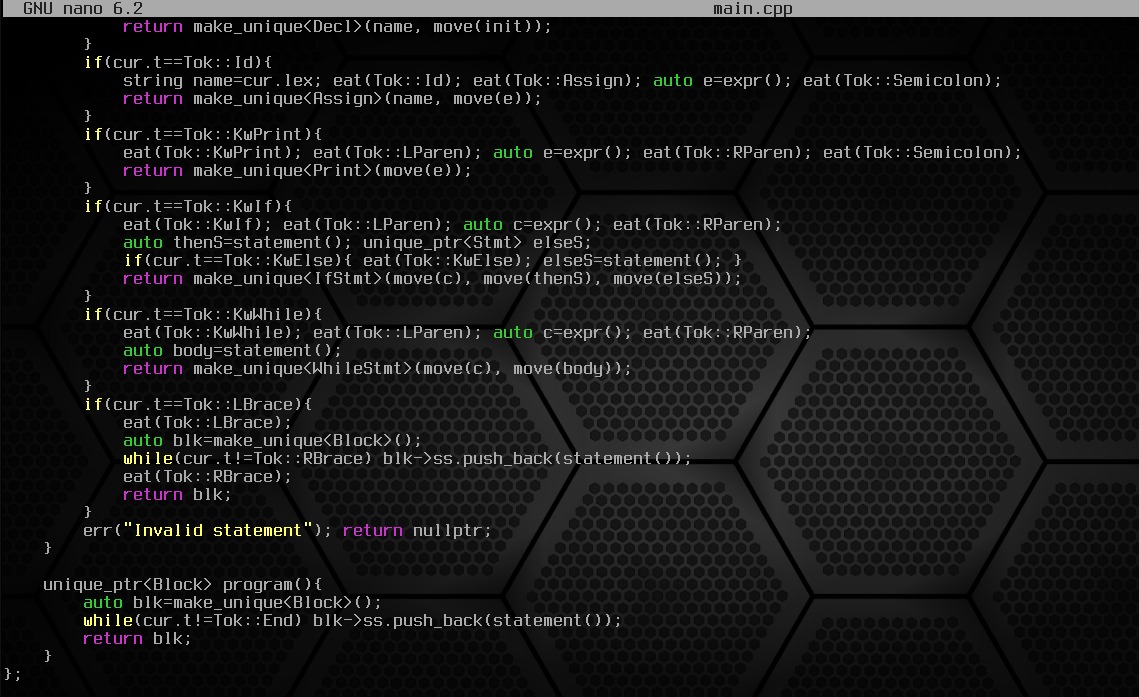
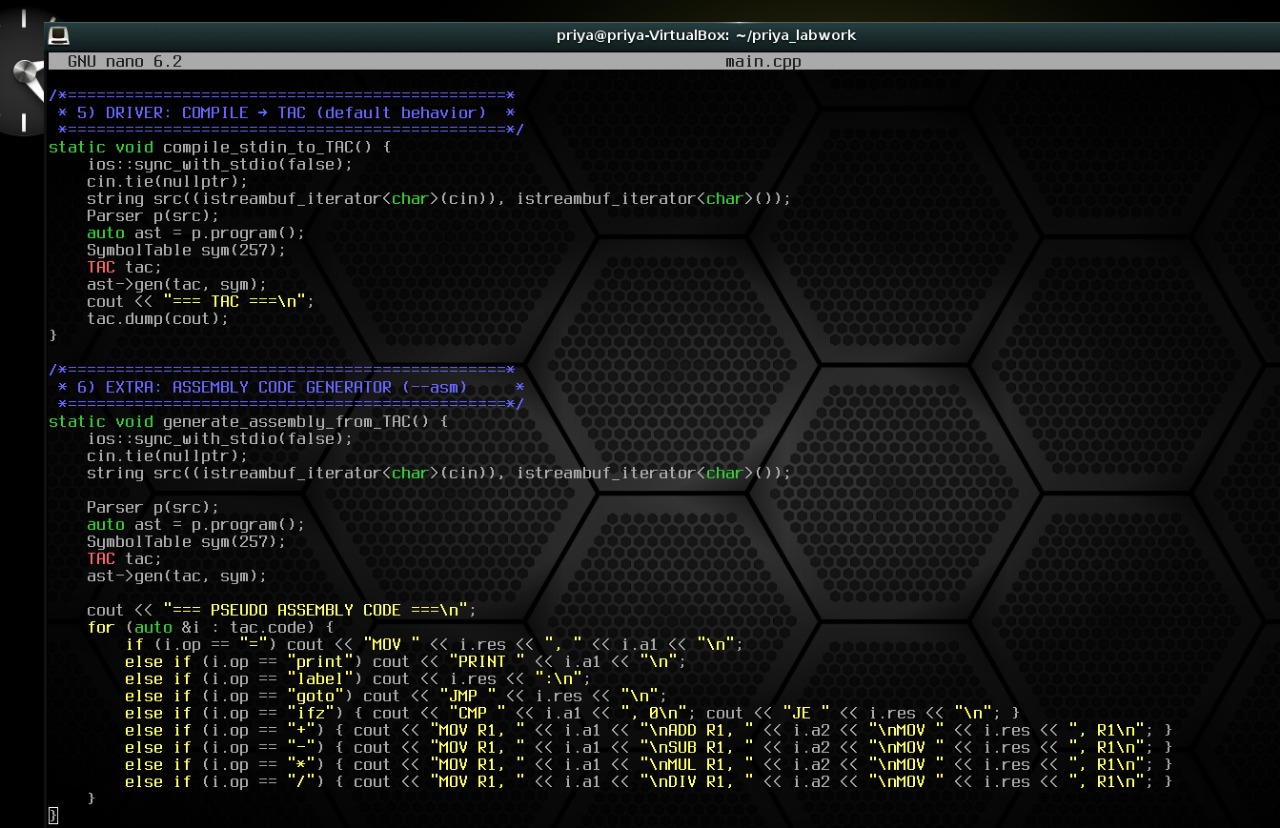
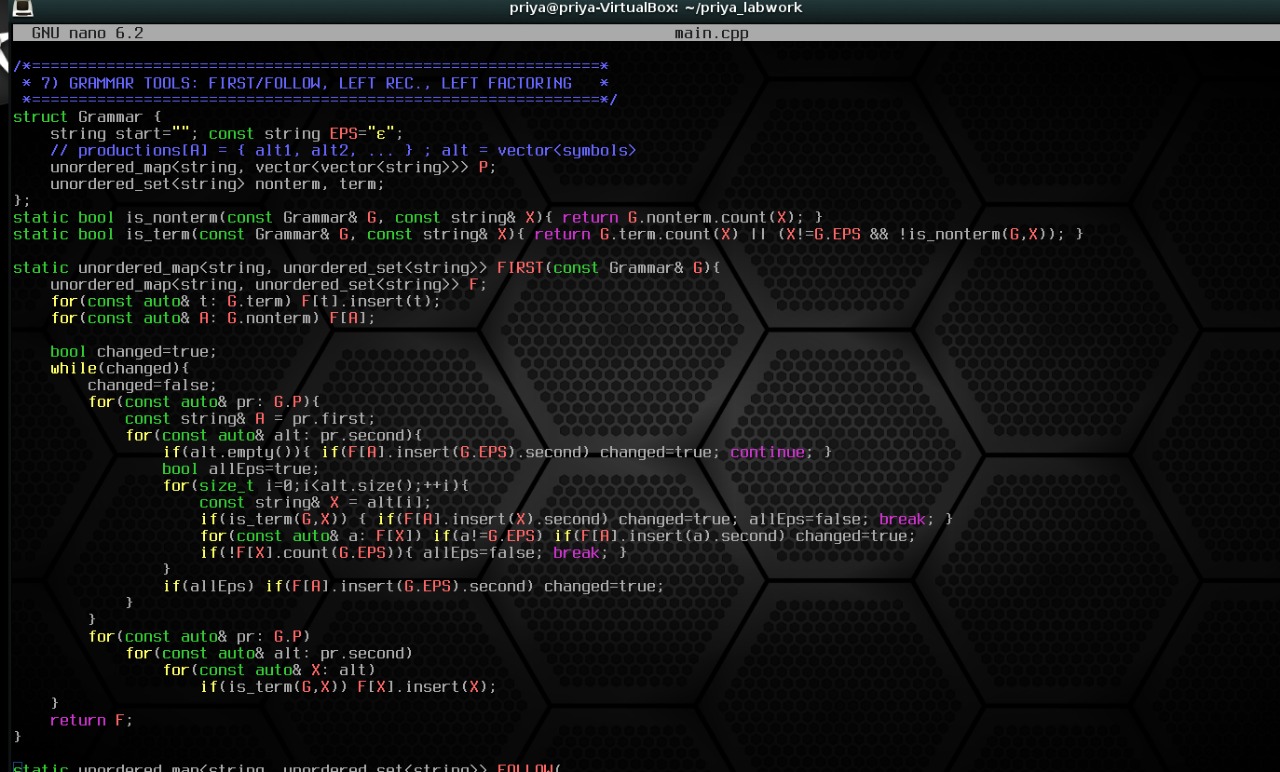
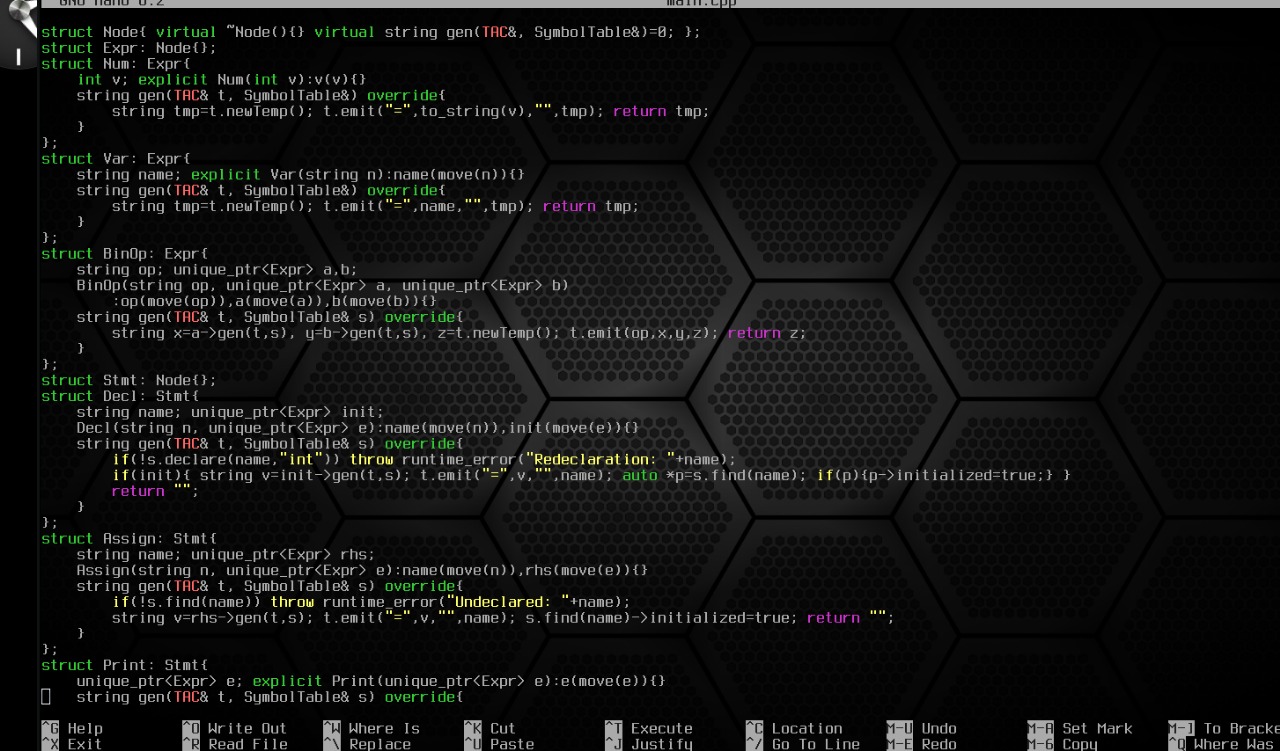
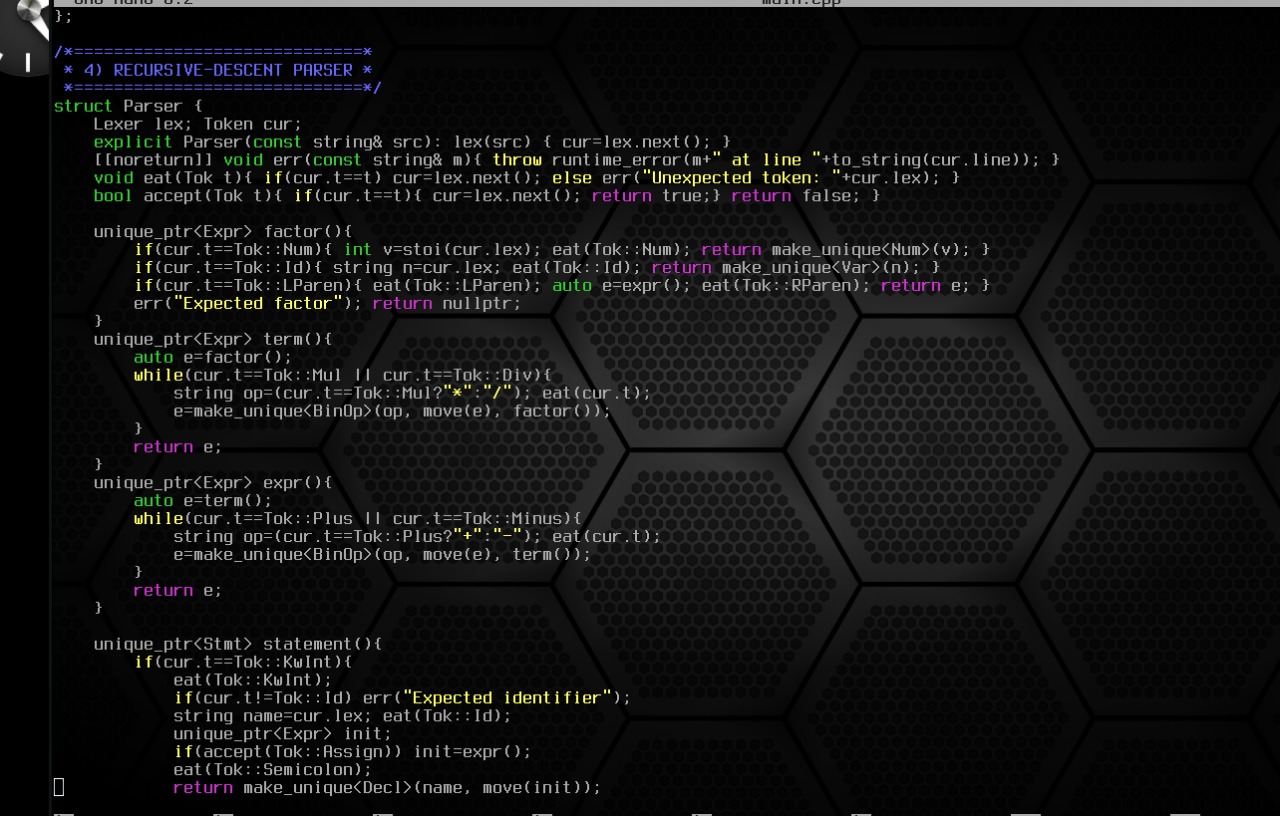
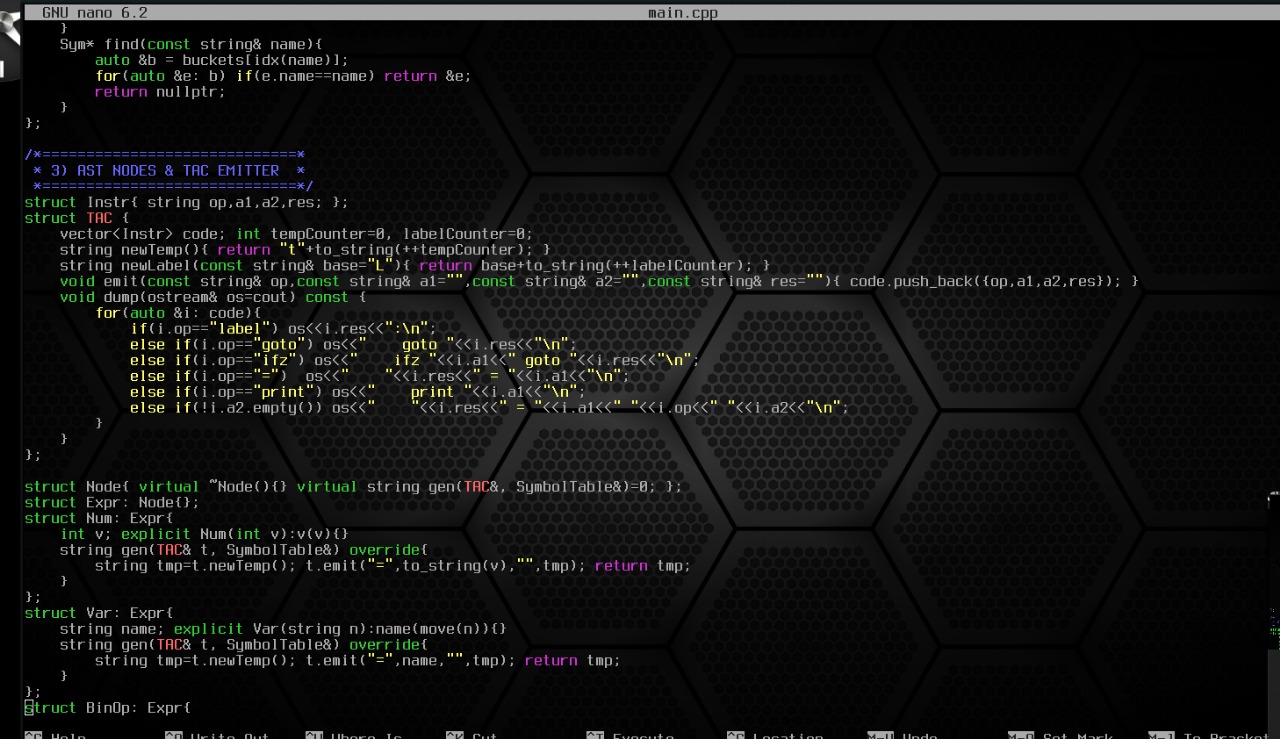
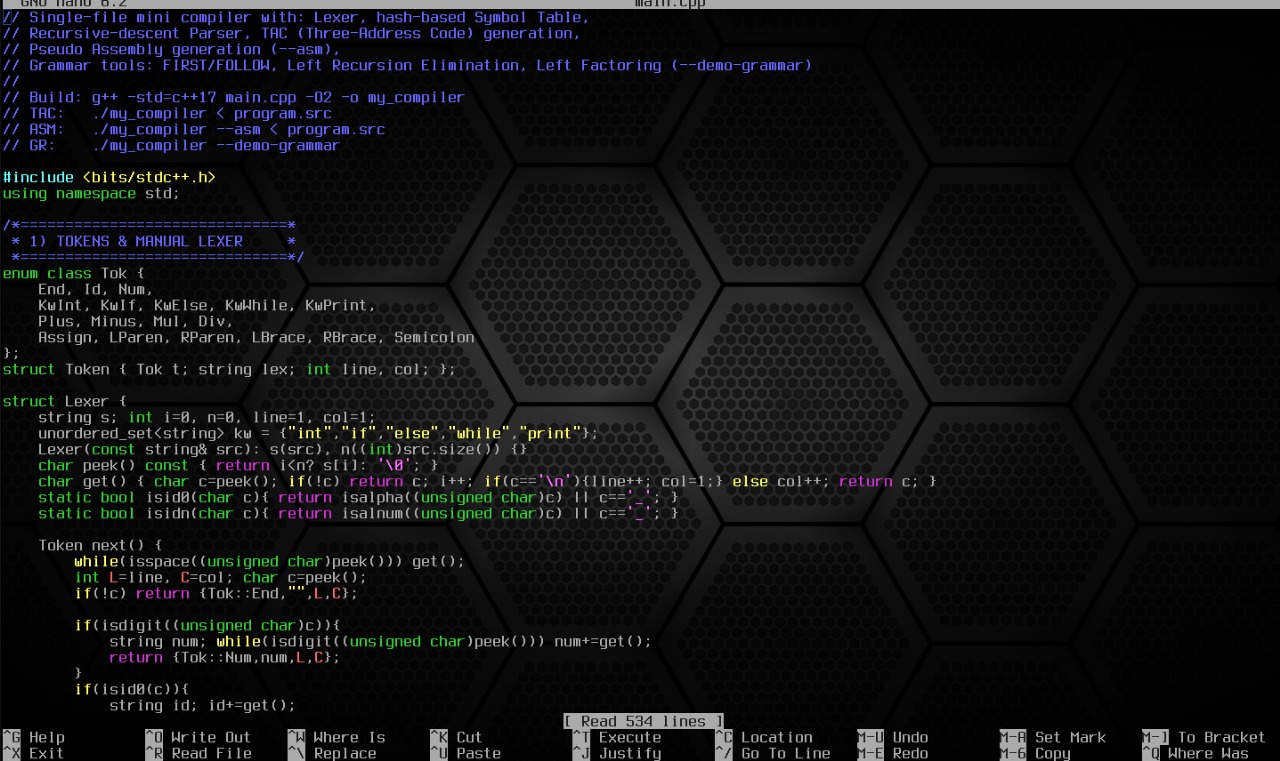
**Section:** D2

**Task:** Create a mini compiler that takes a small amount of code asinput and it generates the machine code after compilation. So, basically, you need to mergeall the phases and apply what you have learned till now to create the compiler. You can usevarious compiler construction tools and use any programming language, including Lex/ YACC.

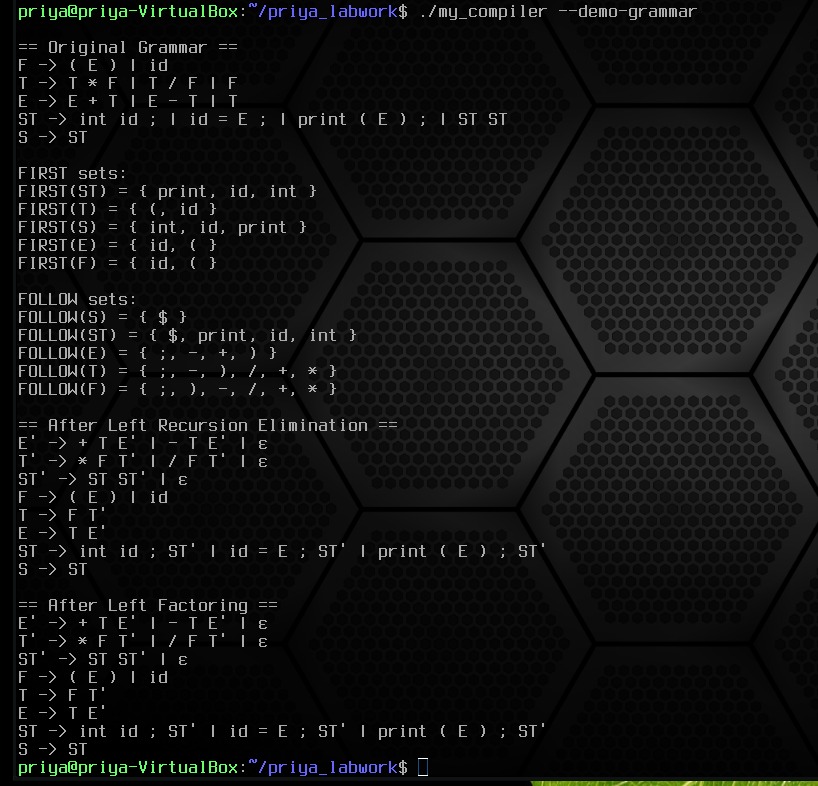
**Block diagram:**

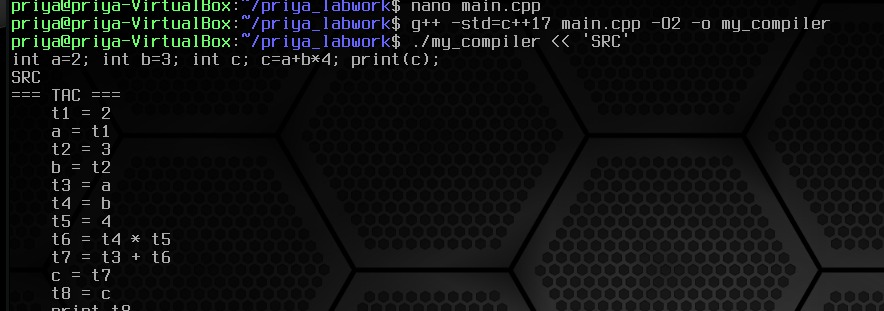
****

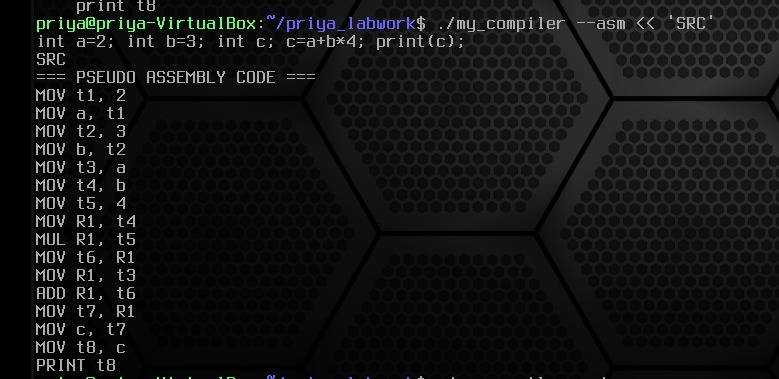
**Main code:**

****

**Input, Output:**







**Implementation:**

The mini compiler was implemented using **C++ (g++ compiler)** on **Bodhi Linux**.  
It is a single-file program (main.cpp) that combines all major phases of a compiler.

**Implementation Steps:**

1. **Lexical Analysis:**  
   Implemented a manual lexer to identify tokens like keywords, identifiers, numbers, and operators.
2. **Symbol Table:**  
   A hash-based symbol table (with chaining) stores variable names, types, and values.
3. **Syntax Analysis (Parser):**  
   A recursive descent parser checks the grammar structure and builds an abstract syntax tree (AST).
4. **Intermediate Code Generation (TAC):**  
   The AST is converted into **Three-Address Code (TAC)** for easier optimization and translation.
5. **Assembly Code Generation:**  
   TAC is translated into **pseudo-assembly code** using MOV, ADD, SUB, MUL, DIV, etc.
6. **Grammar Tools:**  
   Implemented FIRST & FOLLOW set calculation, **Left Recursion Elimination**, and **Left Factoring** for grammar analysis.
7. **Execution Modes:**
   * ./my\_compiler → Generates TAC
   * ./my\_compiler --asm → Generates Assembly Code
   * ./my\_compiler --demo-grammar → Shows Grammar Analysis