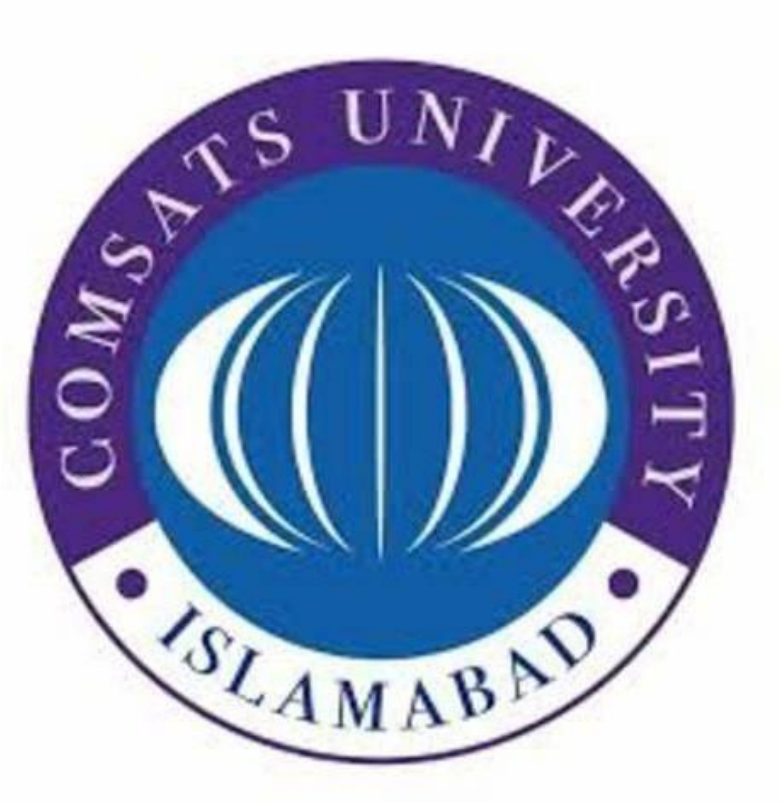
COMSATS UNIVERSITY ISLAMABAD,

ATTOCK CAMPUS

Department Of Computer Science



LAB TERMINAL Q1

Course Name: CC

Submitted To: Mr. BILAL

Submitted By: kulsoom bano

Registration No: Sp22-Bcs-059

1. Briefly Explain Your Project

My project is a Mini Compiler built in C# that implements all the fundamental phases of compilation for a simplified subset of the C programming language. The goal is to simulate how a real compiler processes source code from start to finish, providing a deep understanding of compiler construction.

The project supports the following compilation stages:

- **Lexical Analysis:** It uses regular expressions to break the input code into tokens such as keywords (int, float, if, print), identifiers, numbers, operators (+, -, =), and symbols (;, {, }), skipping whitespaces and comments.
- **Syntax Analysis:** It parses the token stream using custom rules to check whether statements follow correct grammar. Supported constructs include variable declarations, assignments, print statements, and if-else blocks. Errors like missing semicolons or unmatched parentheses are reported clearly.
- **Semantic Analysis:** This phase checks for meaning and context errors. It verifies variable declarations before use, checks for type mismatches (e.g., assigning a float to an int), and maintains a symbol table containing variable names, types, scope, and values.
- **Intermediate Code Generation (TAC):** It converts each valid statement into Three Address Code, a simple format used for further processing. For example, a = b + c; becomes:

  ini

  t1 = b + c

  a = t1
- **Code Optimization:** Basic optimizations are applied to the TAC. This includes constant folding (2 + 3 becomes 5) and eliminating redundant operations (x = x + 0 becomes x = x).

- **Target Code Generation:** It generates a pseudo-assembly version of the optimized code using instructions like MOV, ADD, and PRINT, simulating how real compilers produce machine-level instructions.

  The project is interactive, menu-driven, and designed to be modular so that each phase can be tested independently. It allows users to see how raw code is transformed step by step through a full compilation process, helping students understand compiler design and implementation practically.