COMSATS University Islamabad,

Attock Campus

**Department of Computer Science**



Submitted by:

**Muhammad Hamza (FA21-BCS-058)**

Submitted to:

**DR. Bilal Bukhari**

**Subject:** Compiler Construction

**Lab Terminal**

**Q3 :- Give examples of code optimizations used in your mini compiler.**

1. **Efficient Parsing Algorithms**

- Use LL(1) or LR(1) parsing for simplicity and efficiency, minimizing backtracking.

- Optimize your grammar to remove ambiguities or left-recursion, making parsing faster.

2. **Memoization**

- Implement caching for repetitive syntax constructs to avoid redundant computations.

- Example: Store results of sub-tree parses for reuse when encountering identical sub-trees.

3. **Optimized Tokenization**

- Use regular expressions with `RegexOptions.Compiled` for faster token matching.

- Minimize the number of regular expressions and group similar tokens together.

4.**Error Recovery Mechanisms**

- Implement panic-mode or phrase-level error recovery to continue parsing after encountering an error.

- This prevents redundant analysis while maintaining meaningful feedback.

5.**Abstract Syntax Tree (AST) Simplification**

- Merge nodes with no syntactic significance directly during AST generation.

- Avoid creating nodes for terminal symbols unless required for semantic analysis.

6. **Streamlined Syntax Rules**

- Pre-compute FIRST and FOLLOW sets if you are using predictive parsing.

- Use lookup tables for faster decision-making.

7. **Parallel Processing**

- If analyzing large code files, consider splitting the input and performing parallel parsing on independent parts.

8. **String Interpolation and Optimization**

- Replace `StringBuilder` or concatenation with efficient string operations during error or debug message generation.

Regex identifierRegex = new Regex(@"^[a-zA-Z\_][a-zA-Z0-9\_]\*$", RegexOptions.Compiled);

public bool IsValidIdentifier(string input)

{

return identifierRegex.IsMatch(input);

}

Dictionary<string, ParseResult> memoizationCache = new();

public ParseResult ParseExpression(string input)

{

if (memoizationCache.TryGetValue(input, out var result))

{

return result; // Return cached result

}

result = PerformParsing(input); // Perform actual parsing

memoizationCache[input] = result;

return result;

}