FORMAL LANGUAGES AND AUTOMATA THEORY
PROJECT ASSIGNMENT

REG#: _____ NAME#: _____

COURSE CODE: CS224 INSTRUCTOR: MUHAMMAD SAJID ALI

TOTAL MARKS: 100

Introduction:

In the warm-up assignment, you built a Lexical Analyzer (LA) using **Flex**, which processed C++ code and generated a token stream based on regular expressions.

Now that you're familiar with the concepts and structure of a lexer, it's time to re-implement the same logic **manually in Python, without using any lexer/parser library**.

This exercise will help you understand how tools like Flex work under the hood — and how regular expressions map to code and logic you can write yourself.

Objective

Develop a **Lexical Analyzer in Python** that performs the same task as your previous Flex-based analyzer for a **C++ code**.

Group Formation

Your group should be the same as for the warm-up project.

Build Lexical Analyzer

The analyzer should:

- Read c++ code from a file.
- Identify all valid tokens using **pythons** `re` module. Figure out all the tokens your language allows. Submit the python code file along with example source files.
- Output tokens in the format:

  ```
  token_type | token_value | line_number
  ```

- Save the tokenized output to a file.

Constraints

- You are **not allowed to use external libraries** like `ply, lex, lark`, or other tokenizer/parser tools.
- You **must use the built-in** `re` **module** and implement your own logic to scan, match, and extract tokens.
- Use only standard python libraries.

Sample Input and Output:

- Sample Input

```
int main() {
    float x = 3.14;
    // This is a comment
    if (x > 0) {
        x = x + 1;
    }
    return 0;
}
```

- Sample Output

```
Line 1: Token = int          → Keyword
Line 1: Token = main         → Identifier
Line 1: Token = (            → Separator
Line 1: Token = )            → Separator
Line 1: Token = {            → Separator
Line 2: Token = float        → Keyword
Line 2: Token = x            → Identifier
Line 2: Token = =            → Operator
Line 2: Token = 3.14         → Float Literal
     ...
```

Project Report:

Add the following details to your warmup project report.

1. Group Members
2. Your Lexical Analyzer Name and overview (brief description of approach)
3. Token Types Handled
4. Sample Examples and Outputs for Lexical Analyzer
5. Challenges Faced / What you learned

Submission Format:

Submit your complete project as a single zipped folder named - Project_Assignment_<underscore separated list of reg number of all group members>. The zipped folder must include:

1. lexer.py file (main python file <you may give different file name of your choice>)
2. .cpp test files (used as input)
3. Output files or screenshots demonstrating the analyzer's results
4. Submit your project report separately as pdf, in the same format as used in the warm-up assignment.

Grading:

1. Group Members
2. Lexical Analyzer (60%)
3. Project Report (20%)
4. Viva (20%)
5. Note: marks for the project report and lexical analyzer may be deducted based on performance in the viva