**Name: Muhammad Awais**

**Registration no. : FA20-BCS-020**

**LAB Terminal: Compiler Construction**

**Project Title: Lexical Analysis and First-Follow Set**

***Q.1:*** **Brief of the Project:**

The project is a comprehensive tool for Compiler Design, encompassing both lexical analysis and the computation of First and Follow sets for a given context-free grammar. The primary objectives include accurately tokenizing source code and facilitating the analysis of grammar rules.

***Q.2:*** **Functionalities:**

Functionality 1: **Lexical Analysis**

* **Function Code**:
* Reads an input file ('e1-example.txt') and tokenizes it into keywords, operators, delimiters, identifiers, and numbers.
* Employs sets to identify and eliminate duplicates.
* Presents results with precision.
* **Output**:
* Explicitly showcases the recognized keywords, operators, delimiters, identifiers, and numbers in the input file.

Functionality 2: **First and Follow Set Calculation**

* **Function Code**:
* Accepts a list of production rules for a context-free grammar.
* Strategically calculates and exhibits the First and Follow sets for each non-terminal symbol.
* Ensures removal of redundant entries using sets.
* **Output**:
* Systematically displays the calculated First and Follow sets for each non-terminal symbol in the provided grammar.

***Q3.*** **Input Data and Output**:

* Input

int main() {

int num;

printf("Enter a number: ");

scanf("%d", &num);

if (num >= 0) {

printf("%d is a positive number.\n", num);

} else {

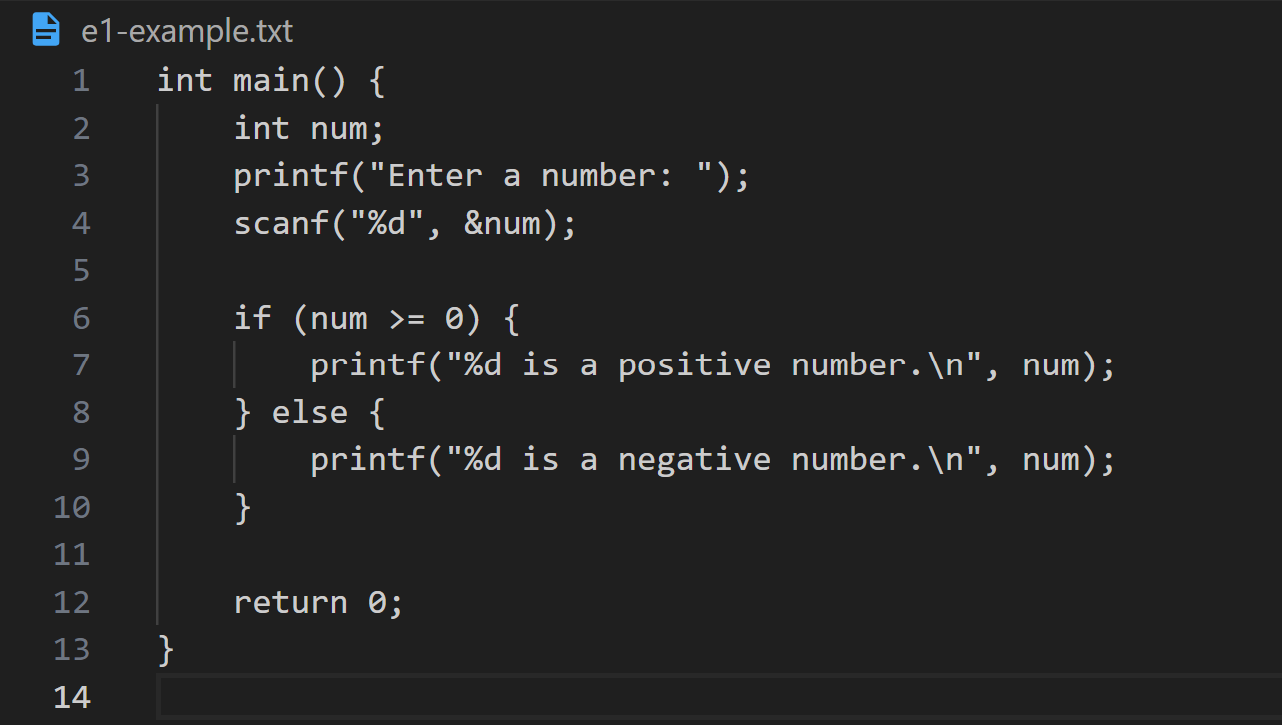
printf("%d is a negative number.\n", num);

}

return 0;

}

**Following is input file**:



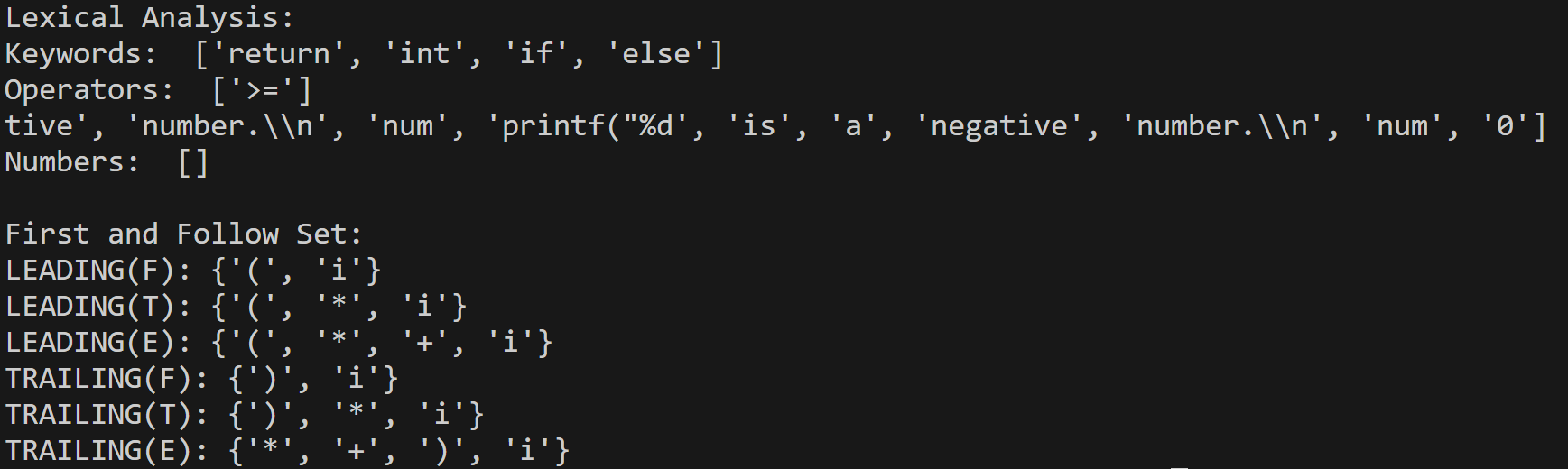
* **Output**:

**Lexical Analysis**

* + **Keywords**: ['return', 'if', 'int', 'else']
  + **Operators**: ['>=']
  + **Delimiters**: ['}', '{']
  + **Identifiers**: ['main', 'num', 'printf("Enter', 'a', 'number:', 'scanf("%d', '&num', 'num', '0', 'printf("%d', 'is', 'a', 'positive', 'number.\\n', 'num', 'printf("%d', 'is', 'a', 'negative', 'number.\\n', 'num', '0']
  + **Numbers**: []

**First and Follow Set**:

* + **LEADING**(F): {'(', 'i'}
  + **LEADING**(T): {'(', '\*', 'i'}
  + **LEADING**(E): {'(', '\*', '+', 'i'}
  + **TRAILING**(F): {')', 'i'}
  + **TRAILING**(T): {')', '\*', 'i'}
  + **TRAILING**(E): {'\*', '+', ')', 'i'}

**Following is the output for given input**:  


***Q4.*** **How Functions work (Step by Step)**:

* **Lexical Analysis**:
  + Open ‘e1-example.txt’ and parse its content.
  + Tokenize the content into distinct categories: keywords, operators, delimiter, identifiers, and numbers.
  + Utilize sets to remove duplicate tokens.
  + Present the identified tokens for each category in a structured manner.
* **First and Follow Set Calculation:**
  + Input a list of production rules representing a context-free grammar.
  + Extract no-terminal symbols from the grammar rules.
  + Methodically compute the First and Follow sets for each non-terminal symbol.
  + Ensure precision by utilizing sets to handle redundancy.
  + Display the calculated sets for analysis and further application.

***Q5.*** : **Challenges Faced During the Project**:

* Managing complex grammars and intricate rules to ensure accurate set calculations.
* Debugging to address potential errors in the code and ensure seamless functionality.
* Coordinating variable scopes and dependencies between different functions within the project.
* Rigorous testing to validate the accuracy of tokenization and set calculations.