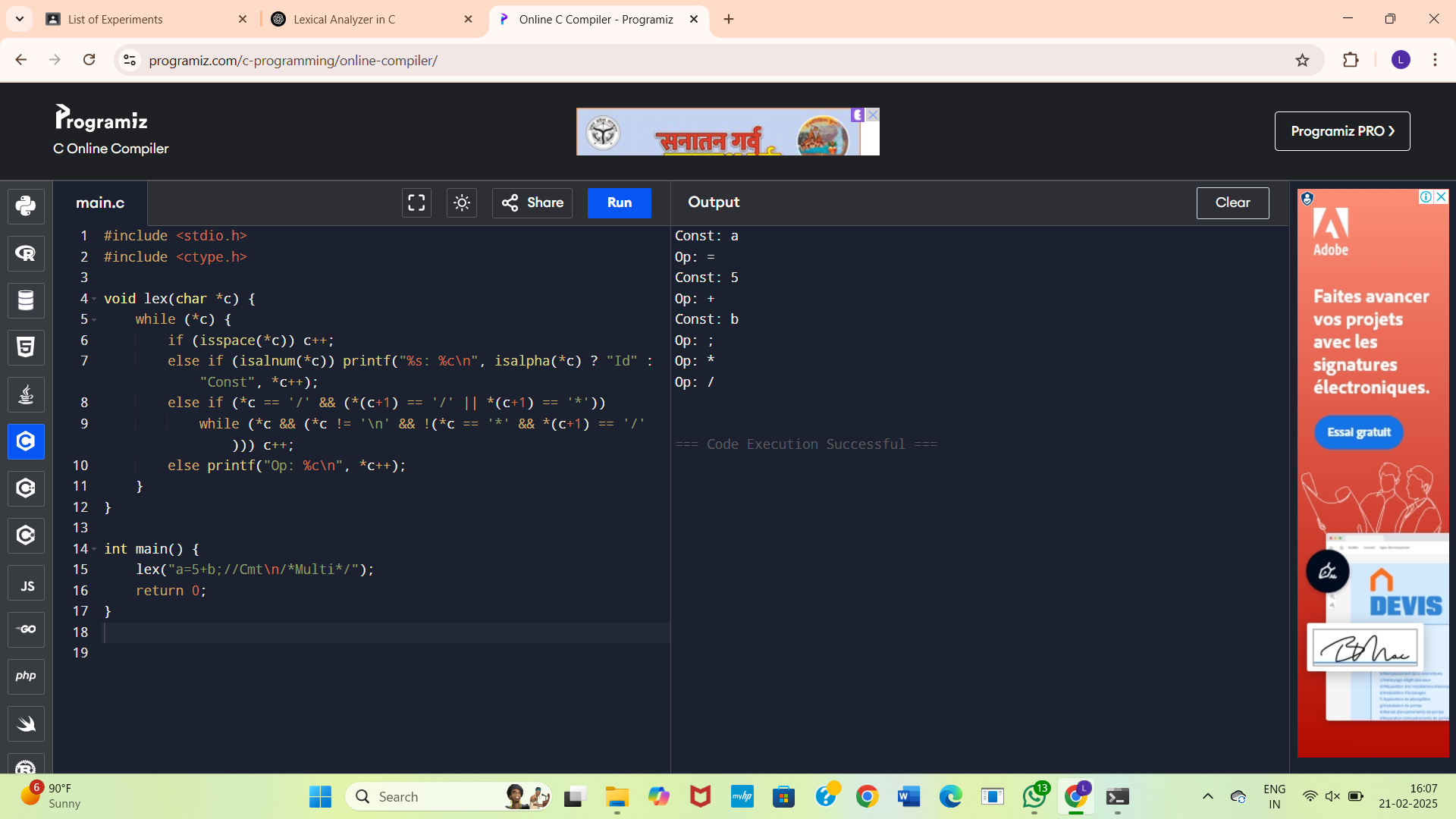
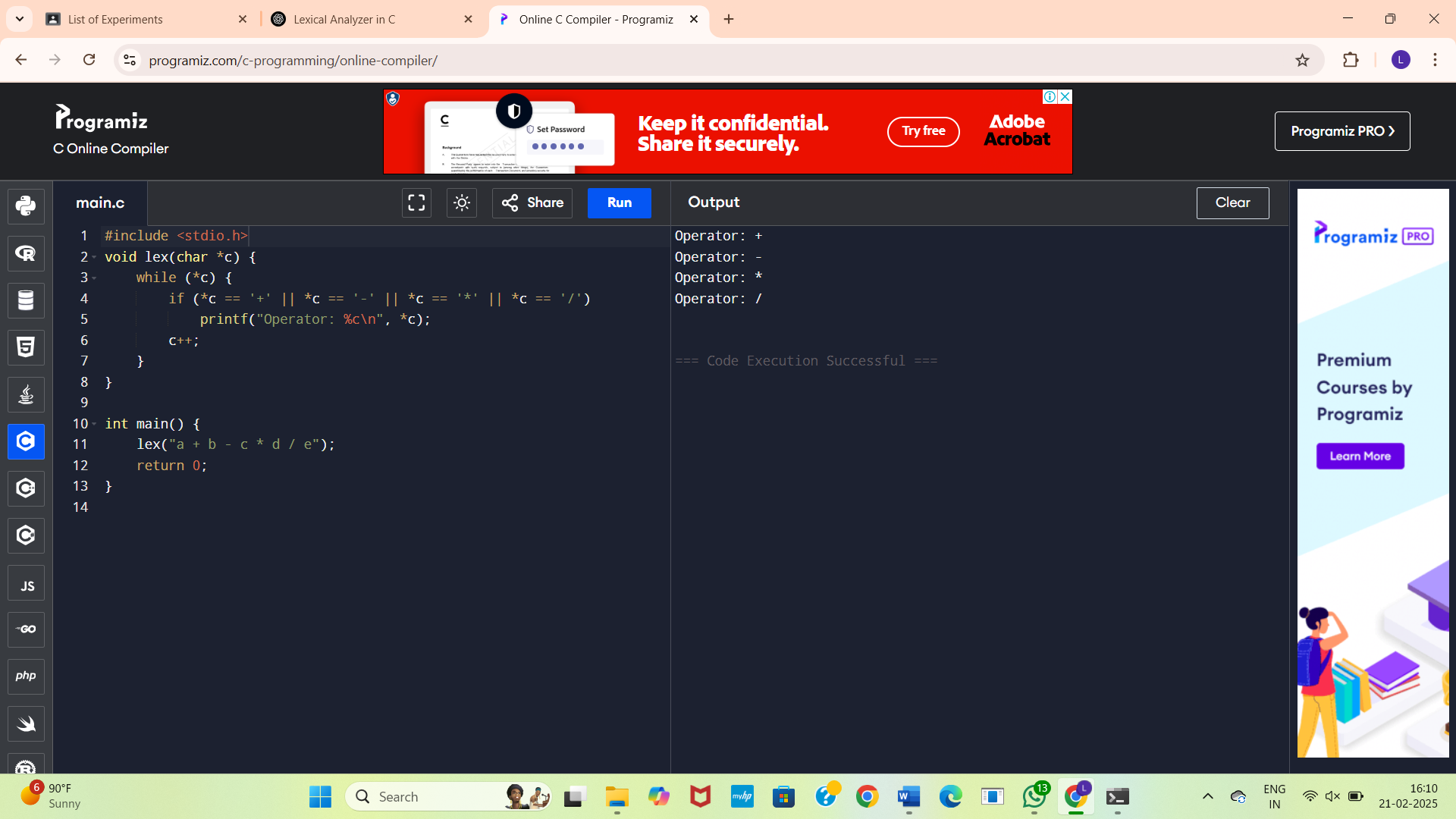
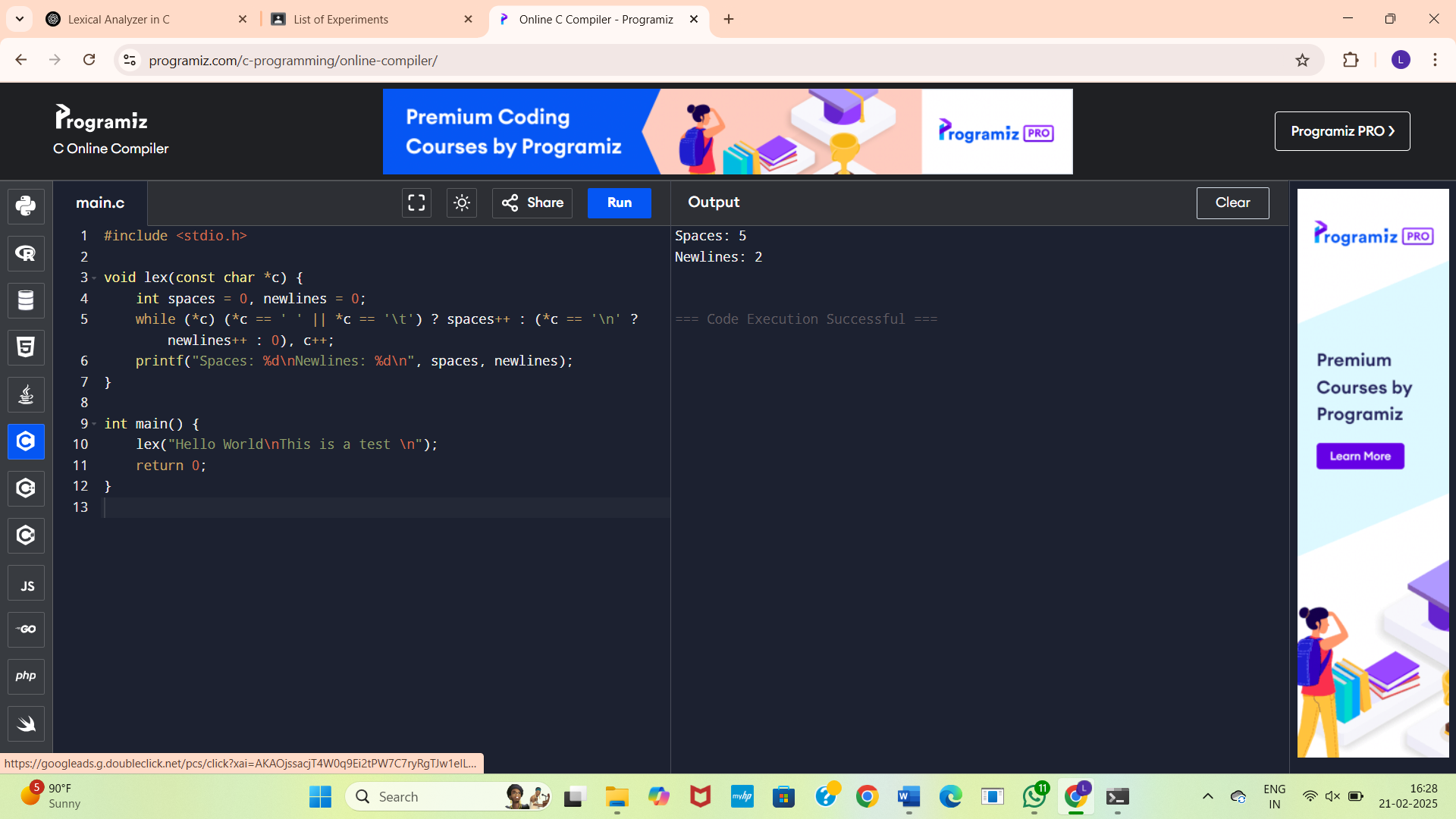
**3. lexical Analyzer**



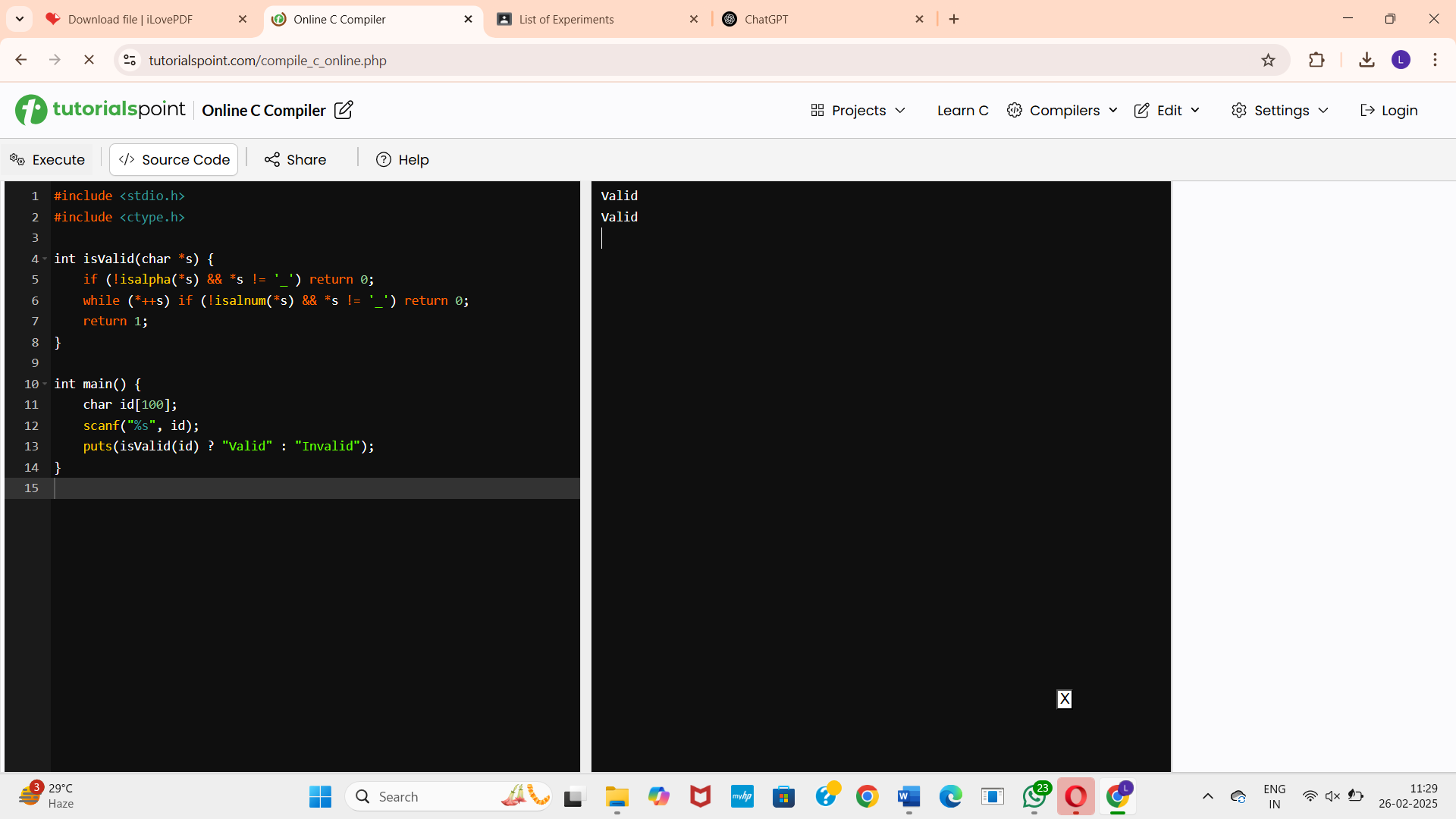
**4. Operator Validator Lexical Analyzer**



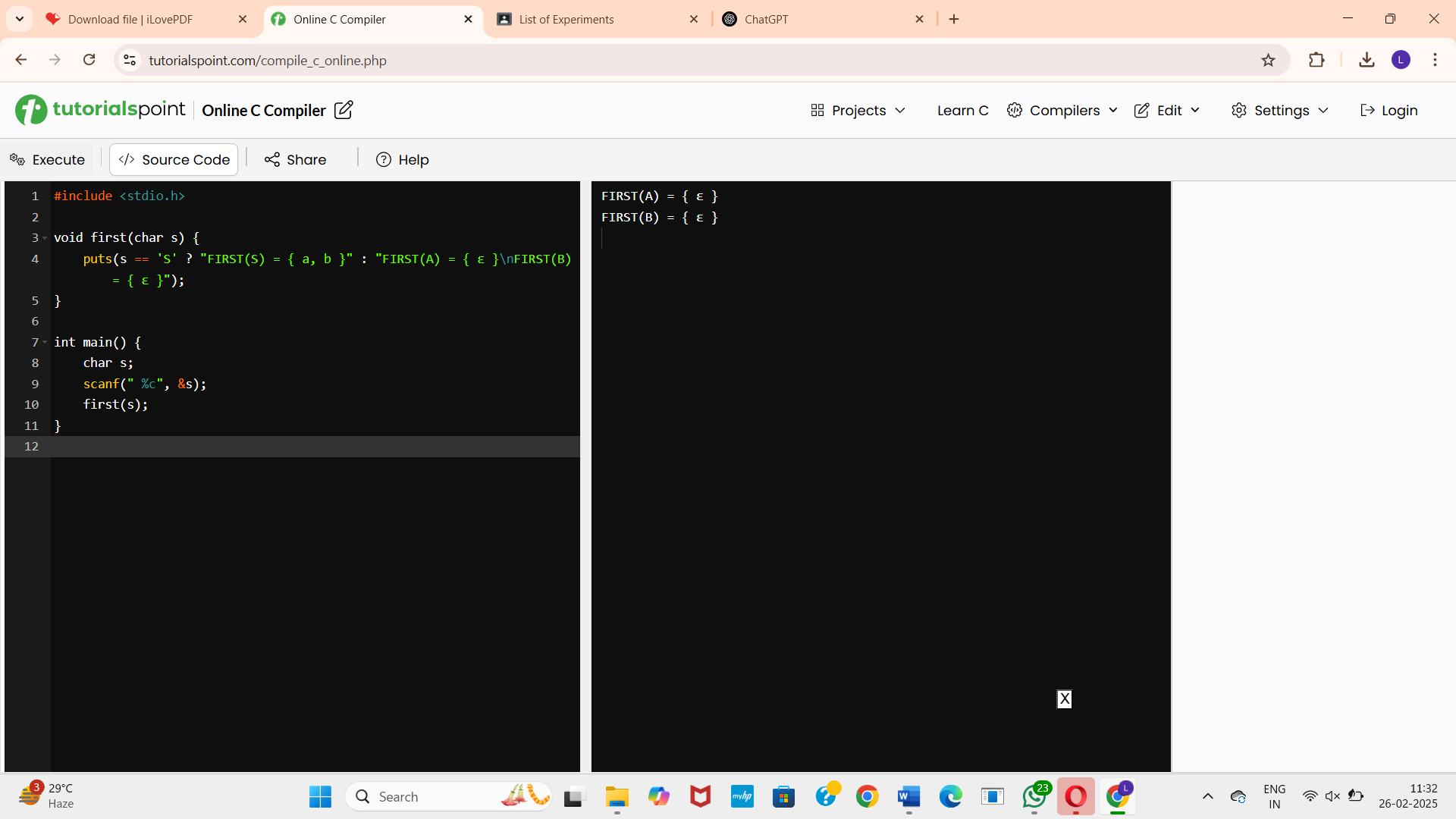
**5.Whitespace and Newline Counter**

****

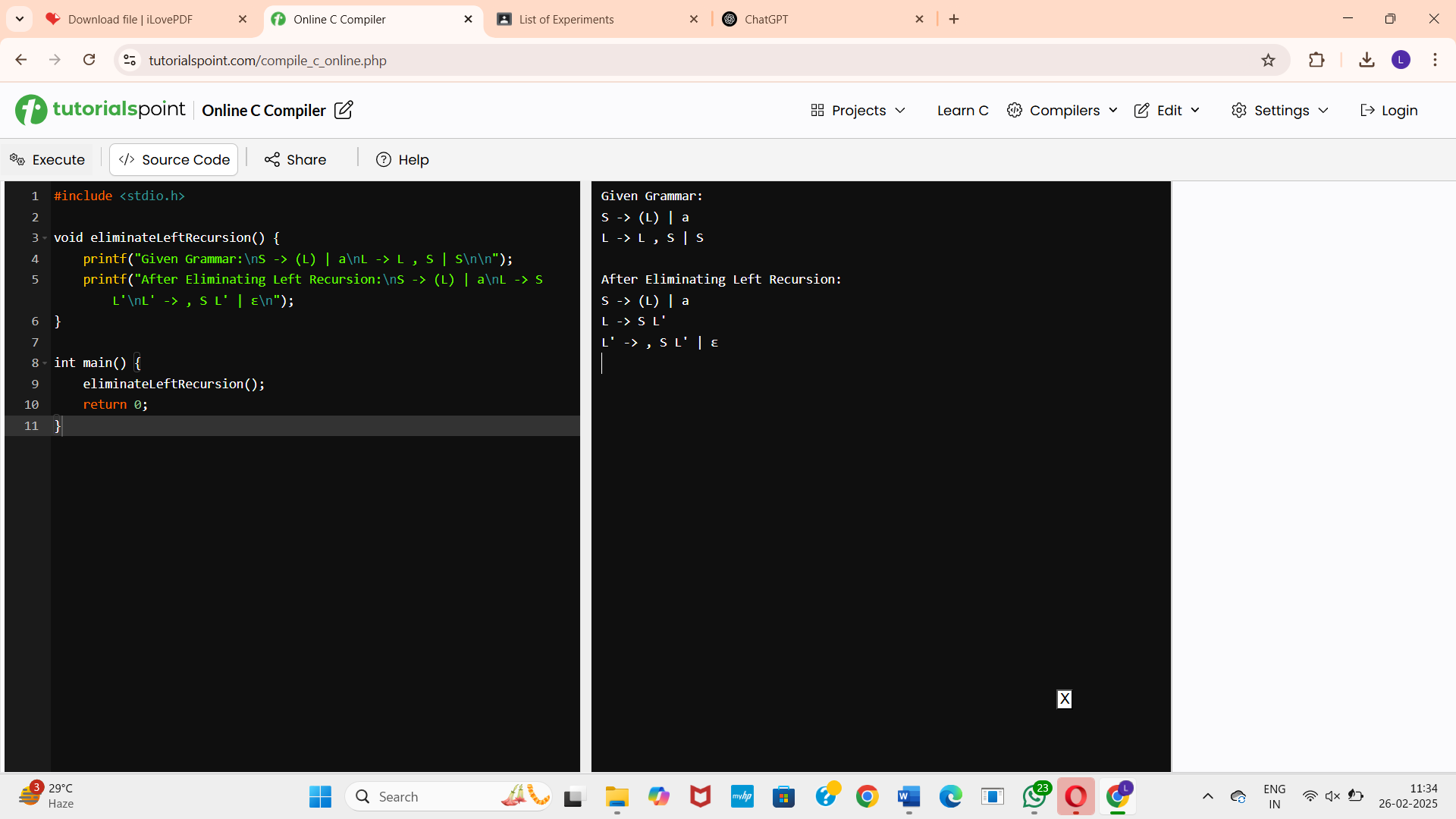
**6.** **lexical Analyzer to test whether a given identifier is valid or not**

****

**8.** **predictive parser for the given grammar**

****

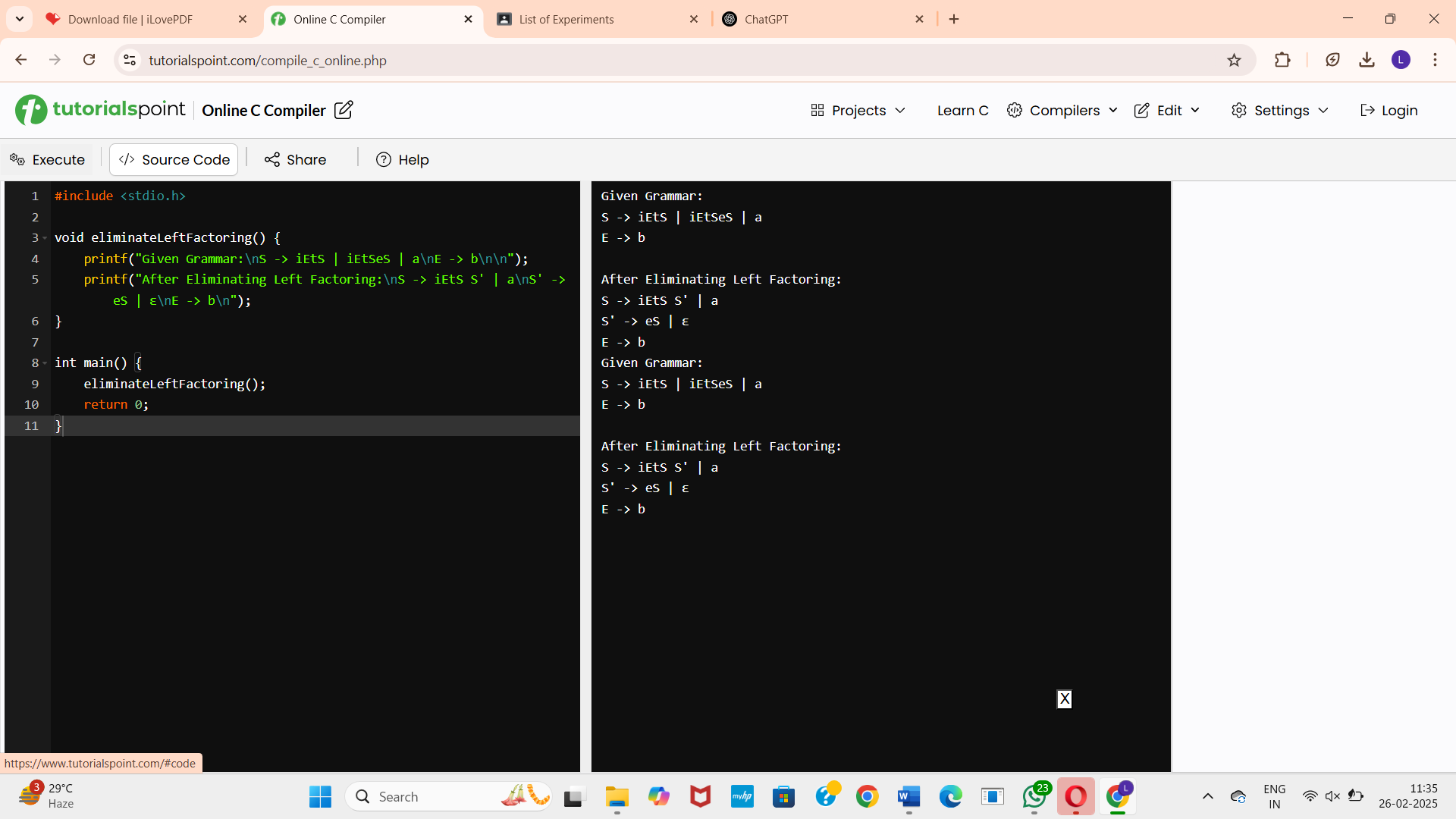
**9.** **to eliminate left recursion from a given CFG**

****

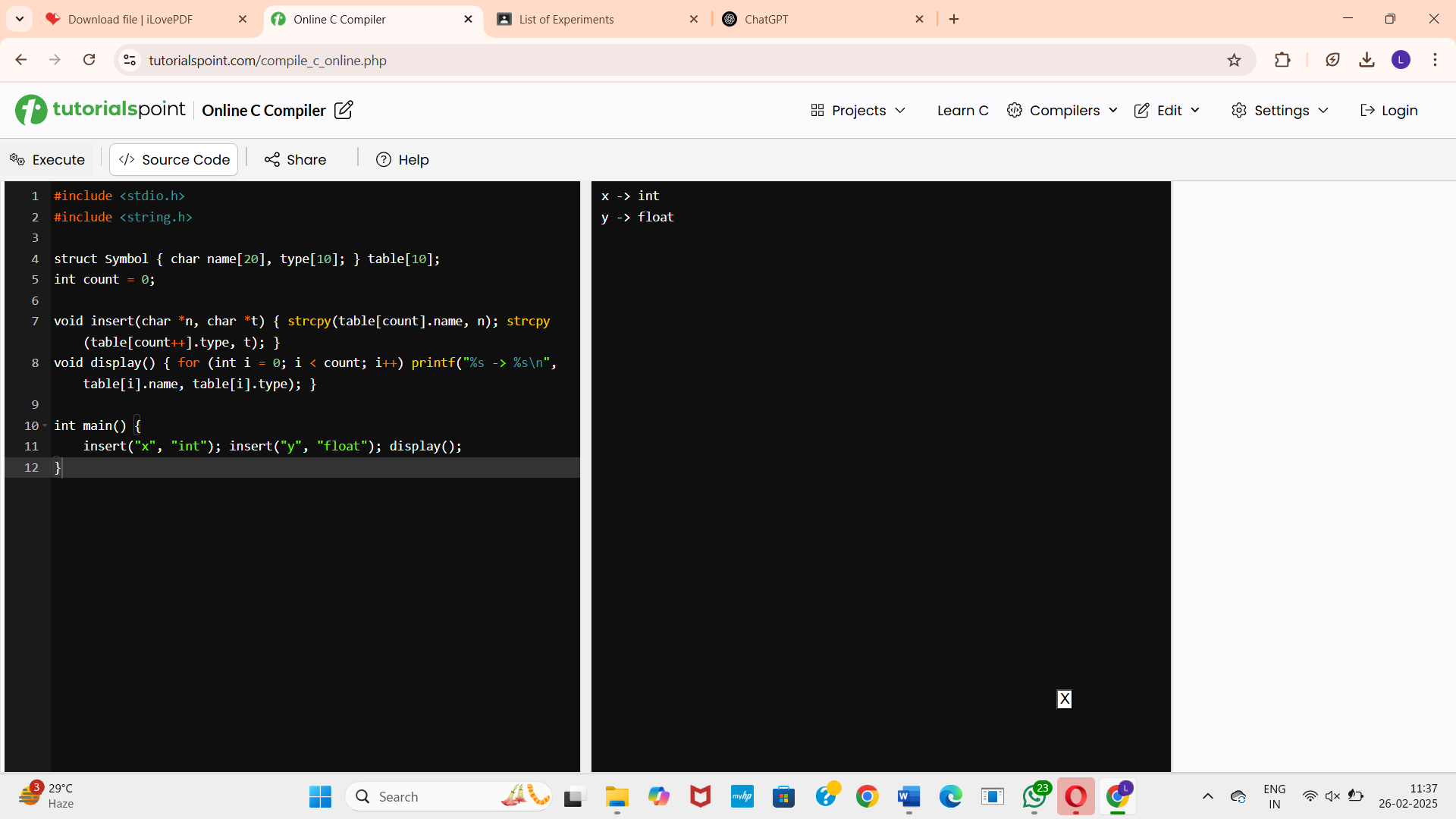
**10.** **program to eliminate left factoring from a given CFG.**

**S → iEtS / iEtSeS / a**

**E → b**

****

**11.** **Implement a C program to perform symbol table operations.**

****

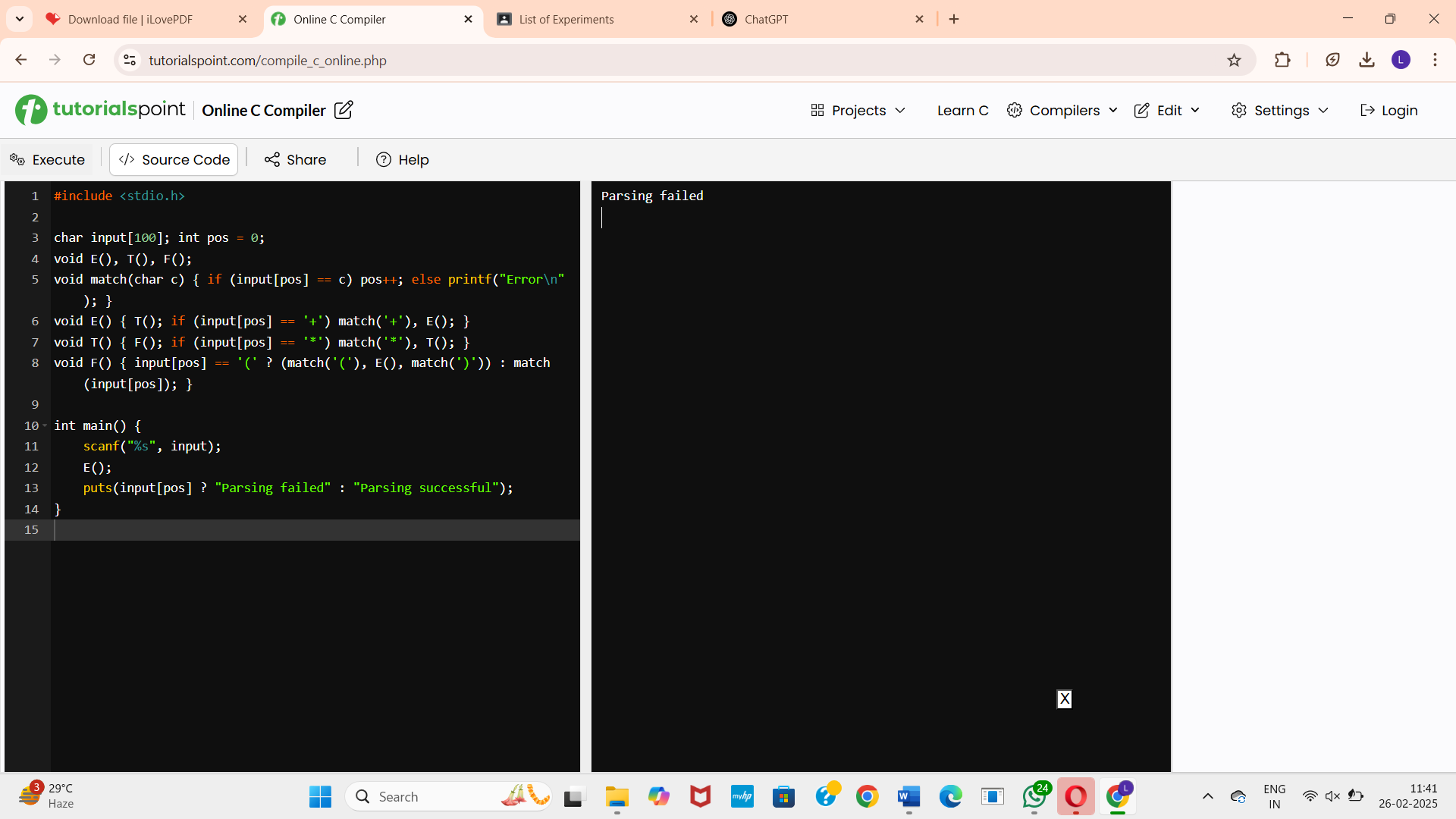
**12.** **Write a C program to construct recursive descent parsing for the given grammar**

**E → TE’**

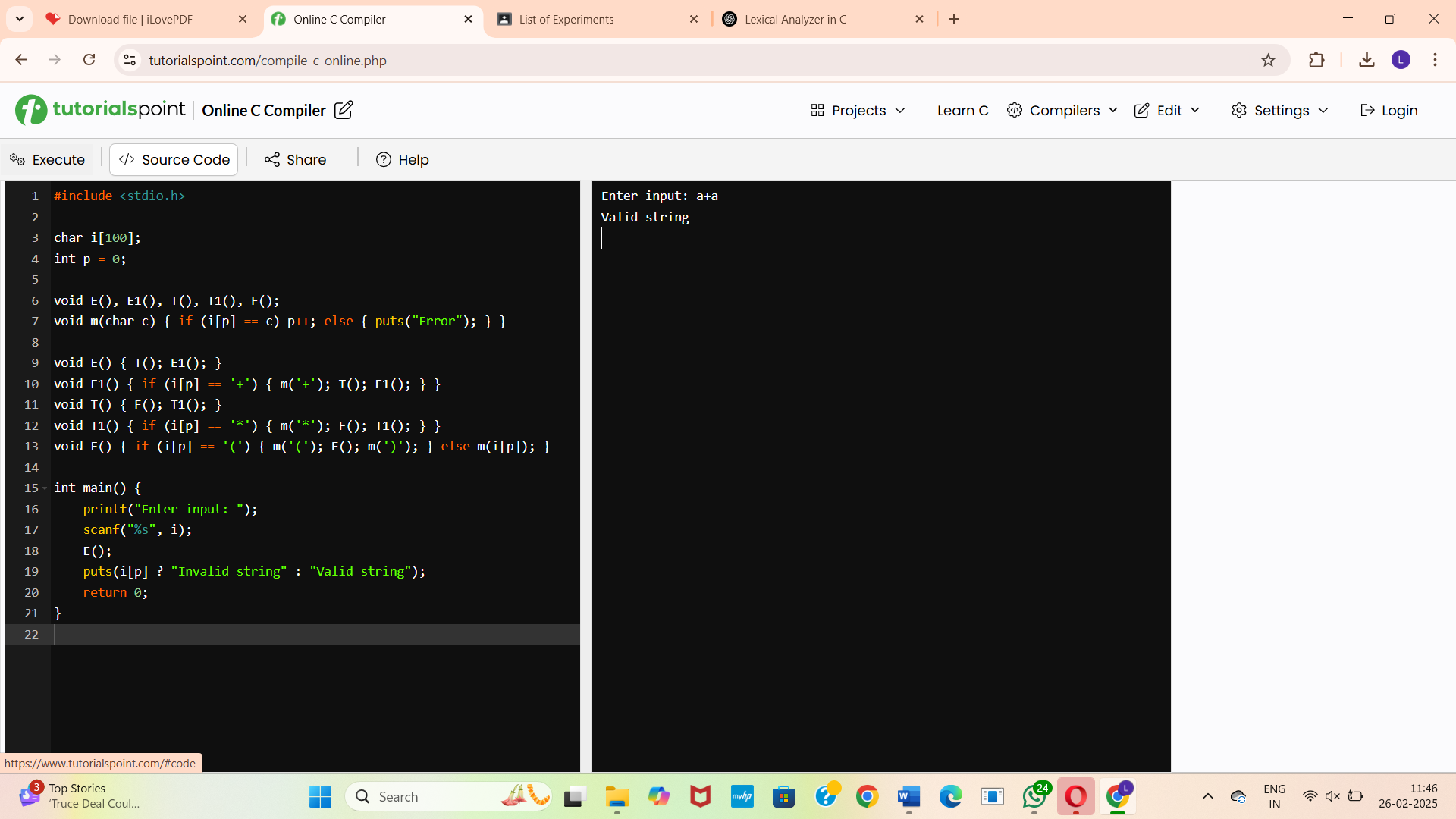
**E’ → +TE’ / ∈**

**T → FT’**

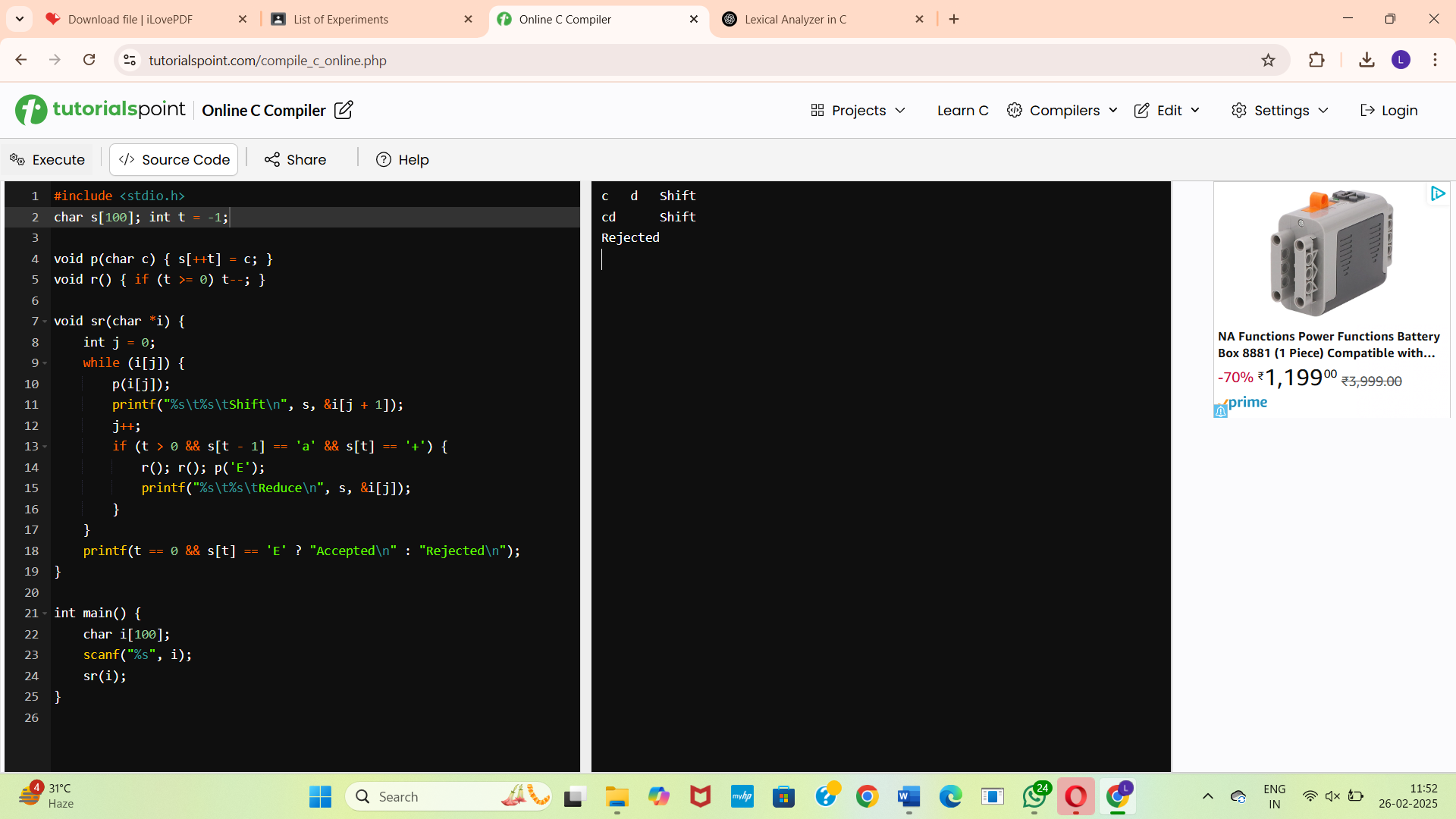
**T’ → \*FT’ / ∈**

****

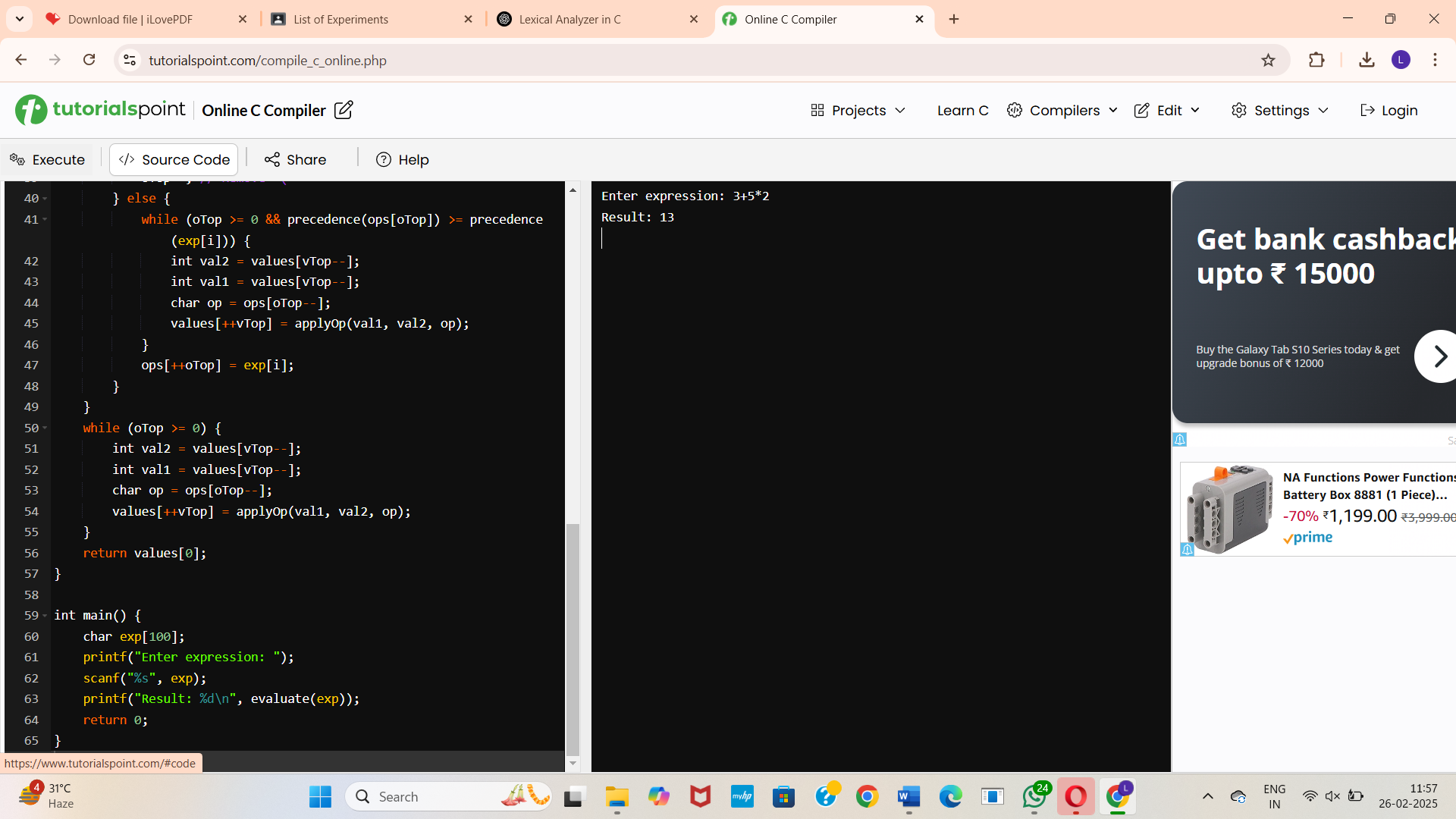
**13. Recursive Descent Parser for Grammar Validation**

****

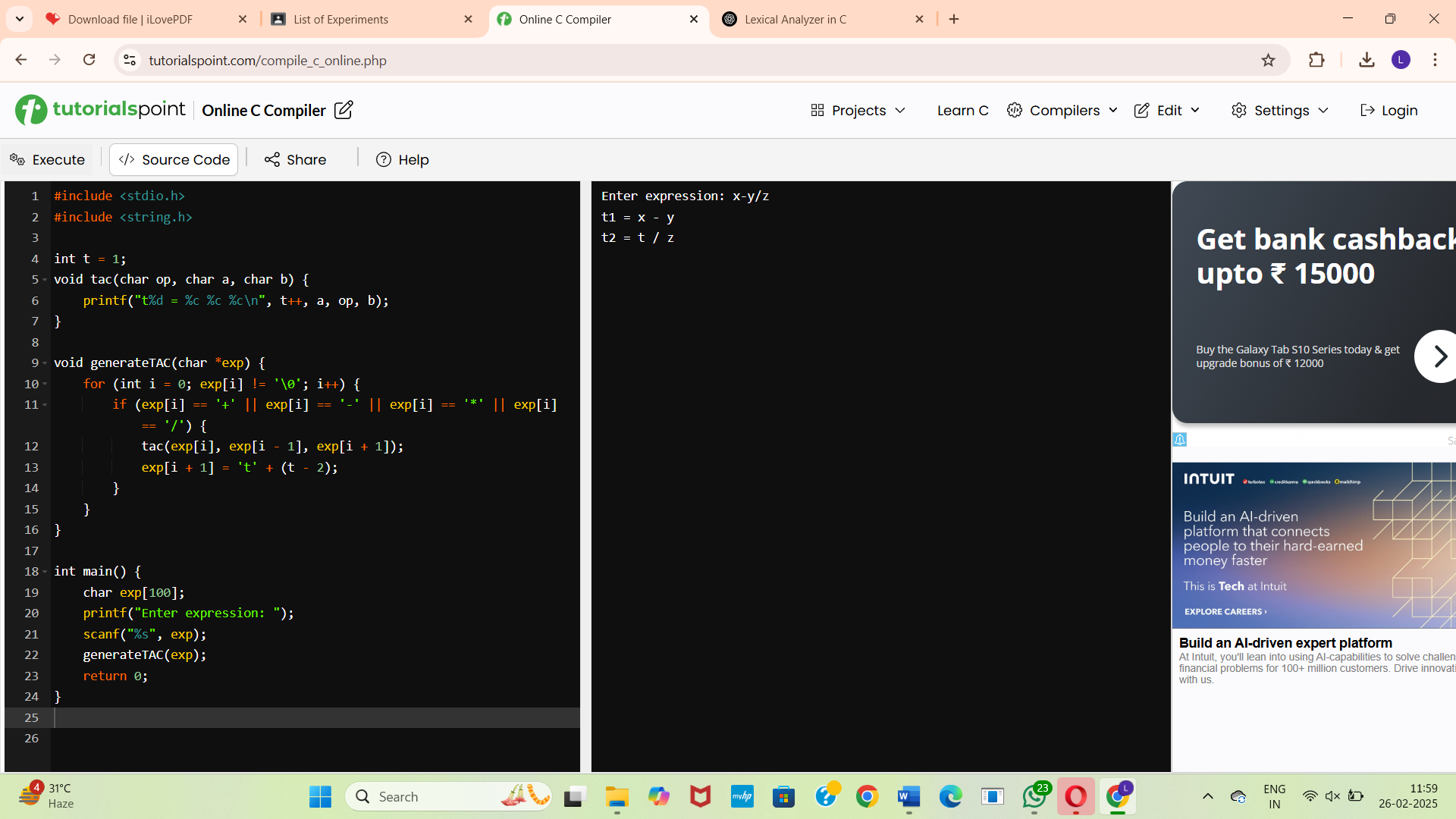
**14.** **Shift-Reduce Parser Implementation in C**

****

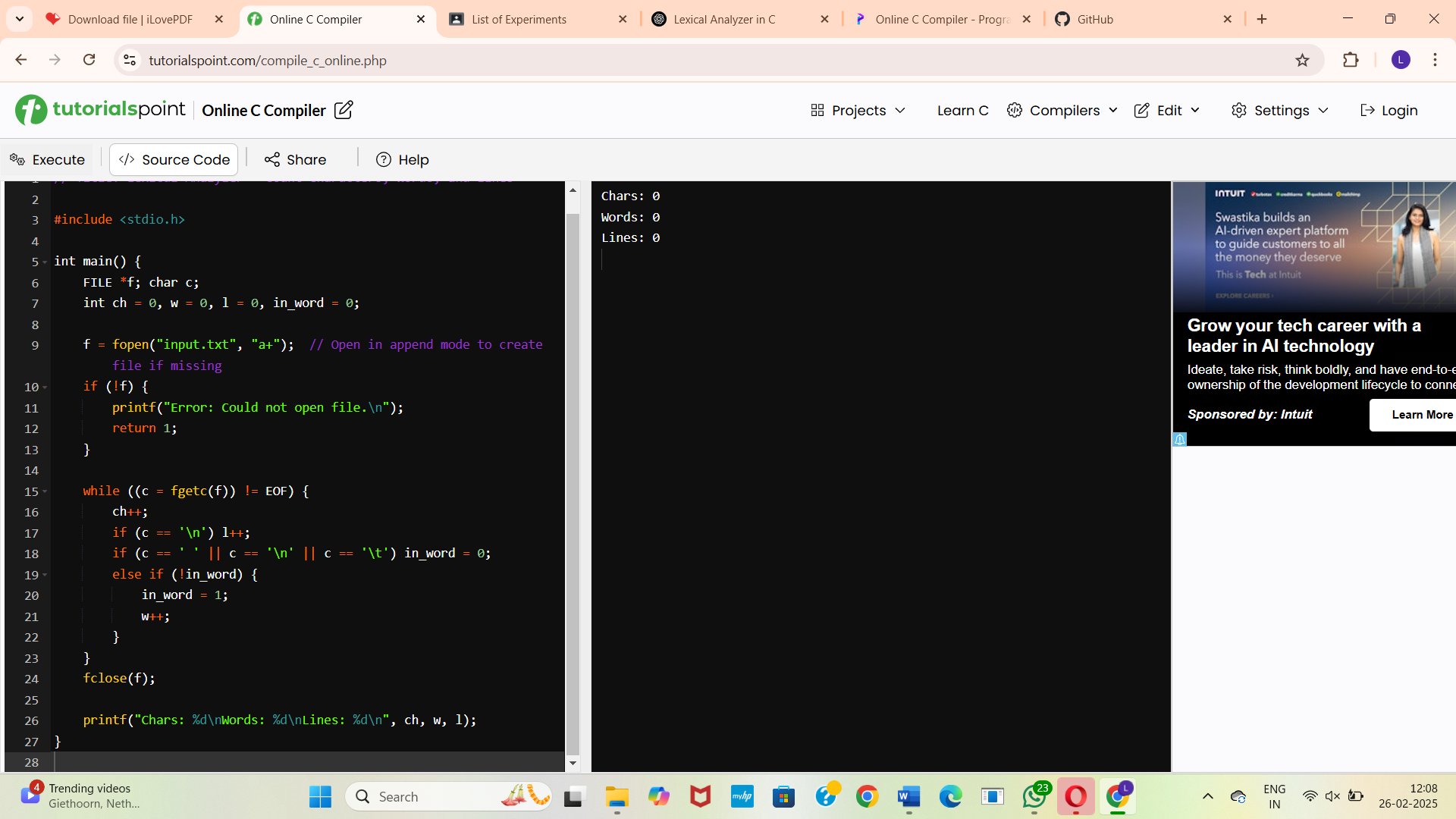
**15.** **Operator Precedence Parsing**

****

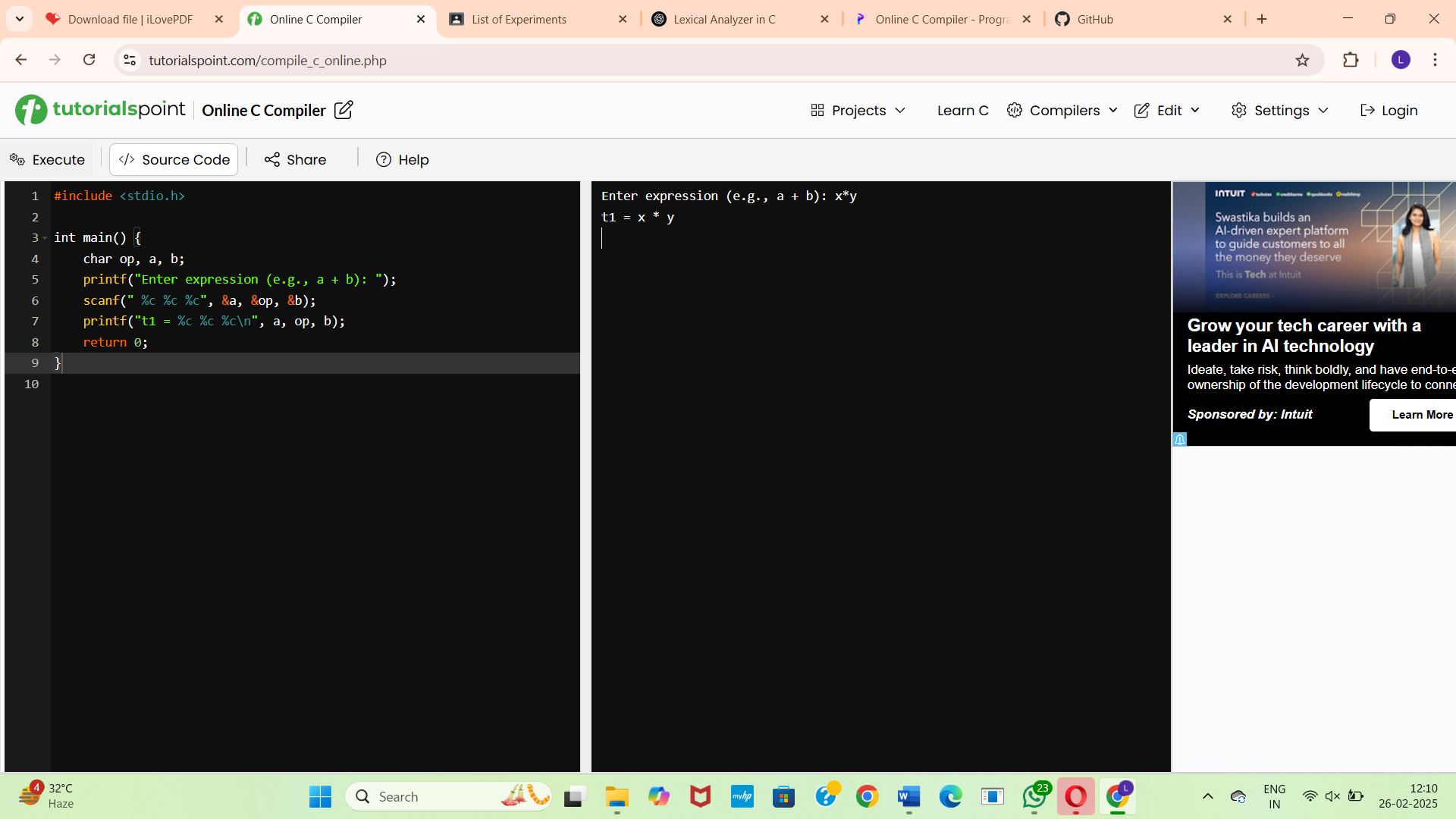
**16. Three Address Code Generation**

****

**17.** **Lexical Analyzer to Count Characters, Words, and Lines in a File**

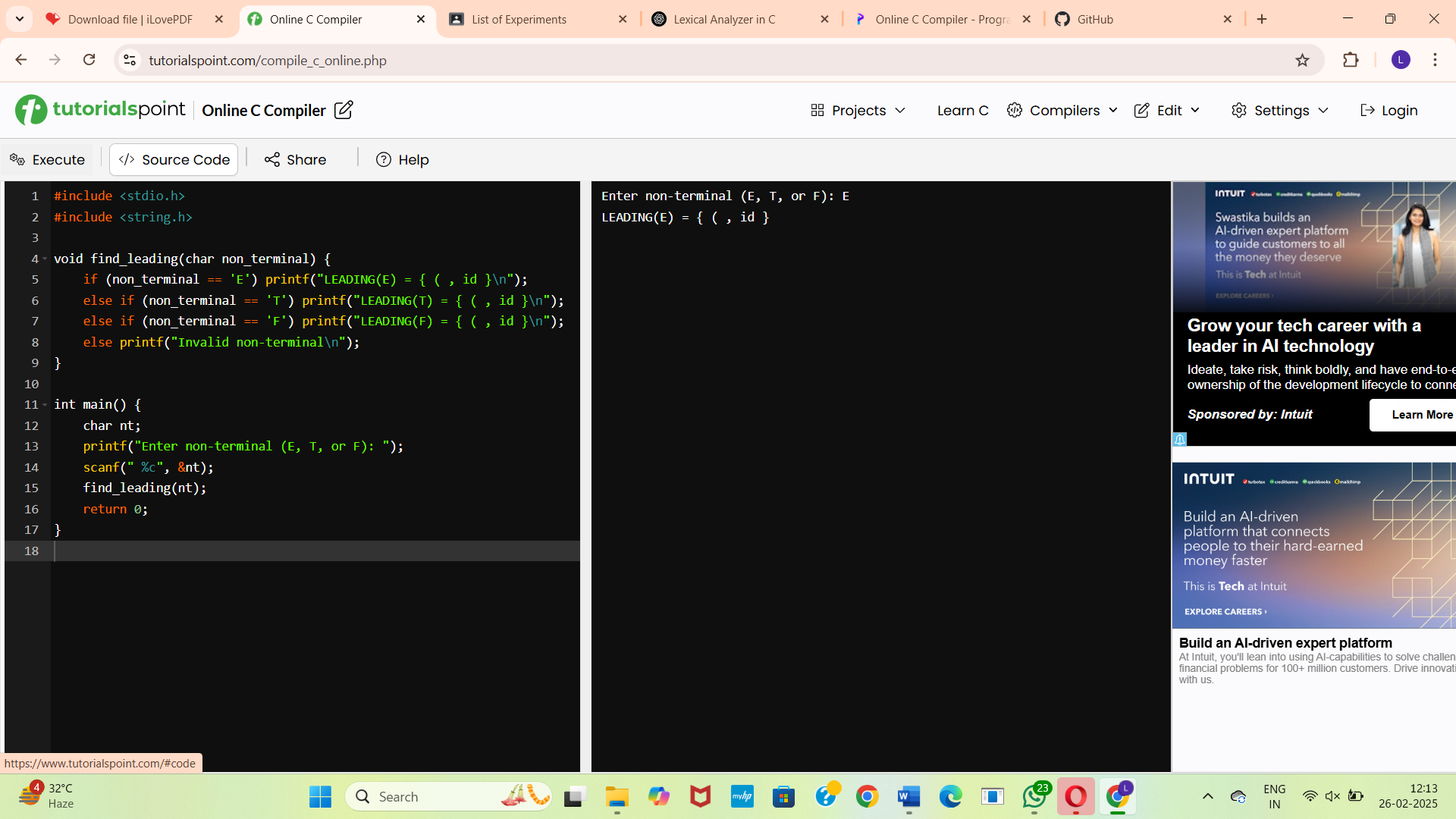
****

**18.** **Write a C program to implement the back end of the compiler.**

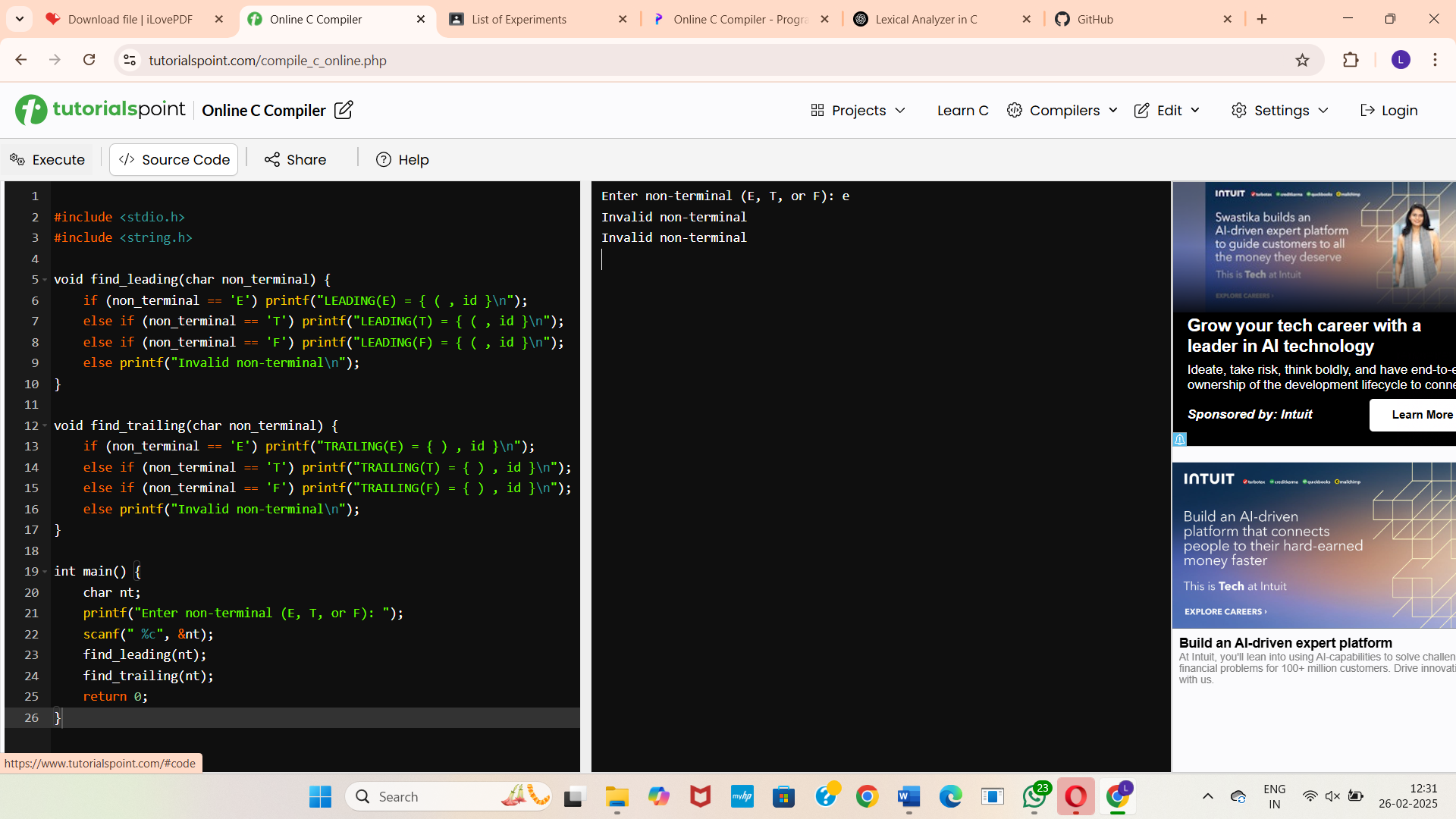
****

**19.**

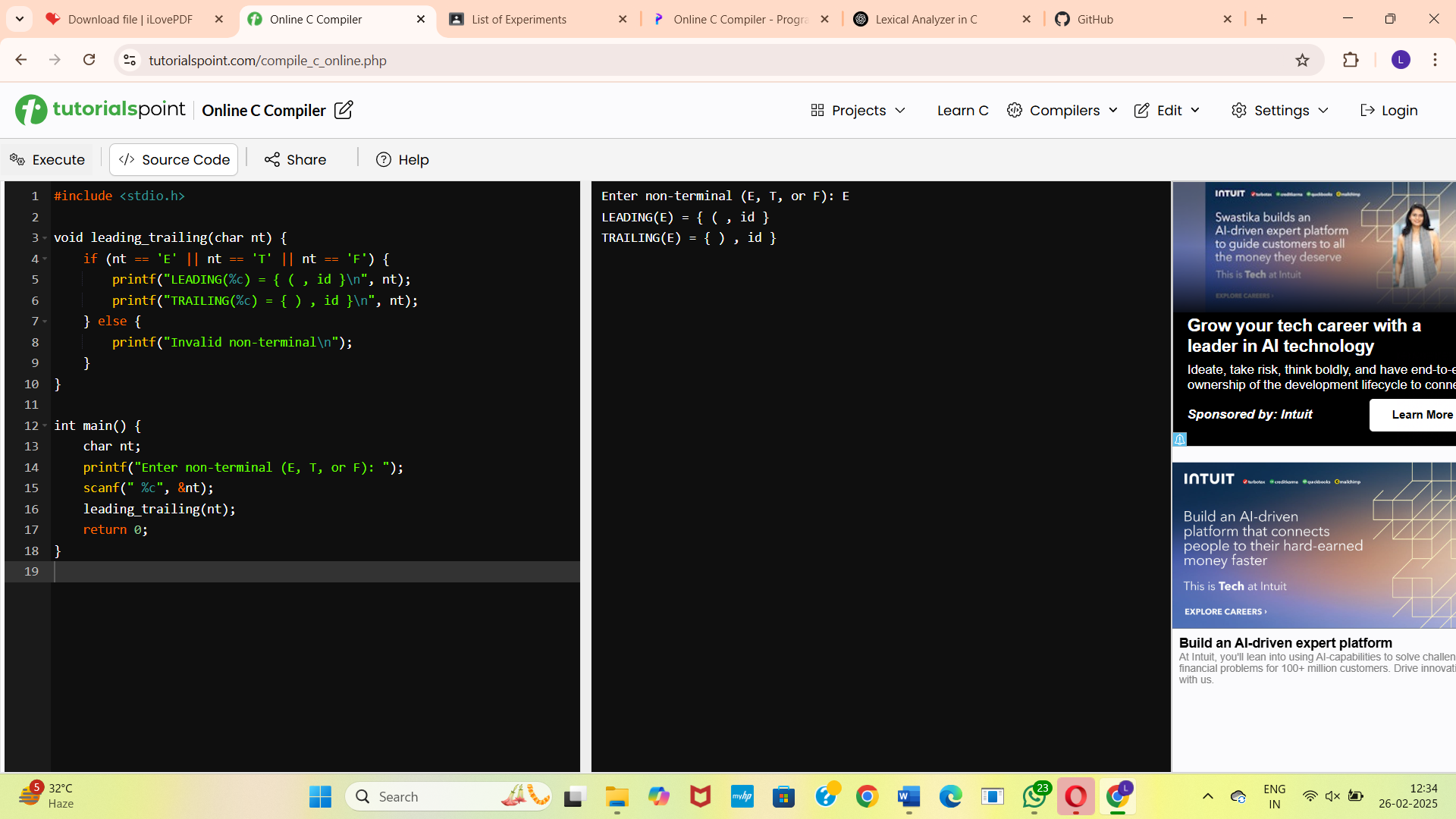
**Compute LEADING() - Operator Precedence Parser"**

****

**20. Compute LEADING() and TRAILING() - Operator Precedence Parser**

****

**21. LEX Program to Count Characters, Words, and Lines in a C Source File**

****