COMPILER DESIGN  
 (19CSE401)  
 LAB – 2

NAME: N SANJANA  
ROLL NO: CH.EN.U4CSE22138

1. AIM: Implementation of Lexical Analyzer Using Lex Tool  
   ALGORITHM:

* Start Lexical Analysis.
* For each token in the input:

If it starts with #, label as preprocessor directive.

If it matches a C keyword (like int, void, etc.), label as keyword.

If it’s a valid name followed by (, label as a function.

If it’s {, label as block begins.

If it’s }, label as block ends.

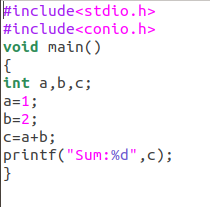
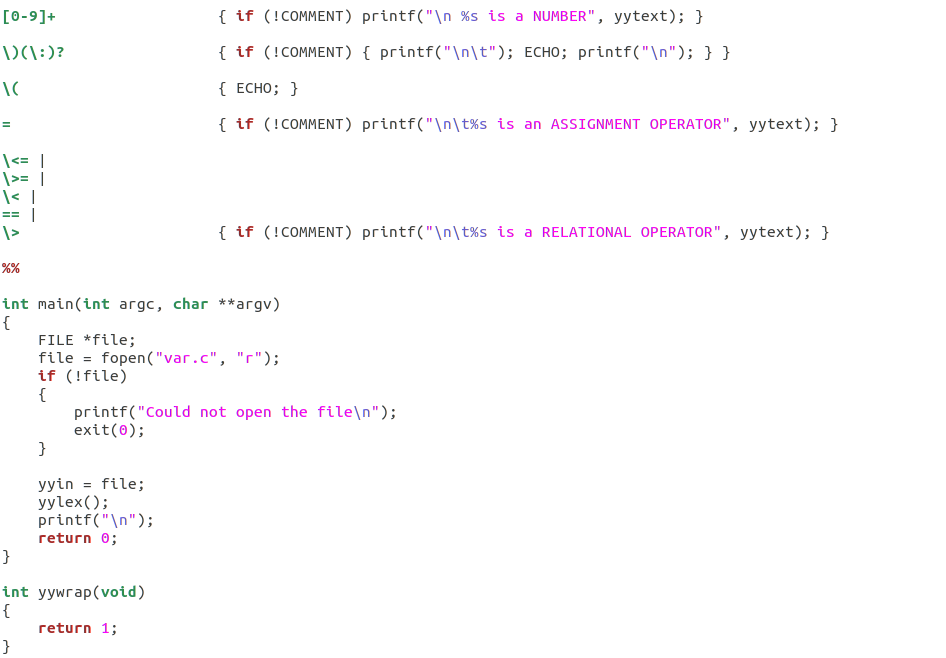
If it's =, label as assignment operator.

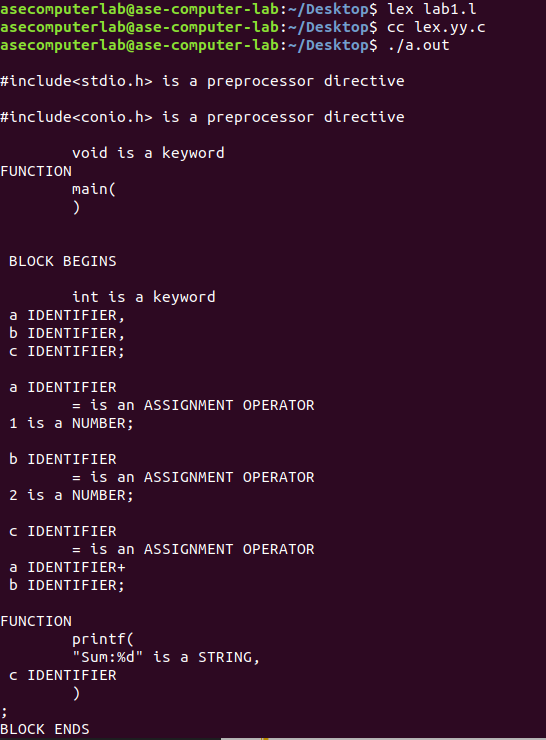
If it's a number ([0-9]+), label as number.

If it's a valid variable name (a-zA-Z[a-zA-Z0-9]\*), label as identifier.

If it's a string (within "), label as string.

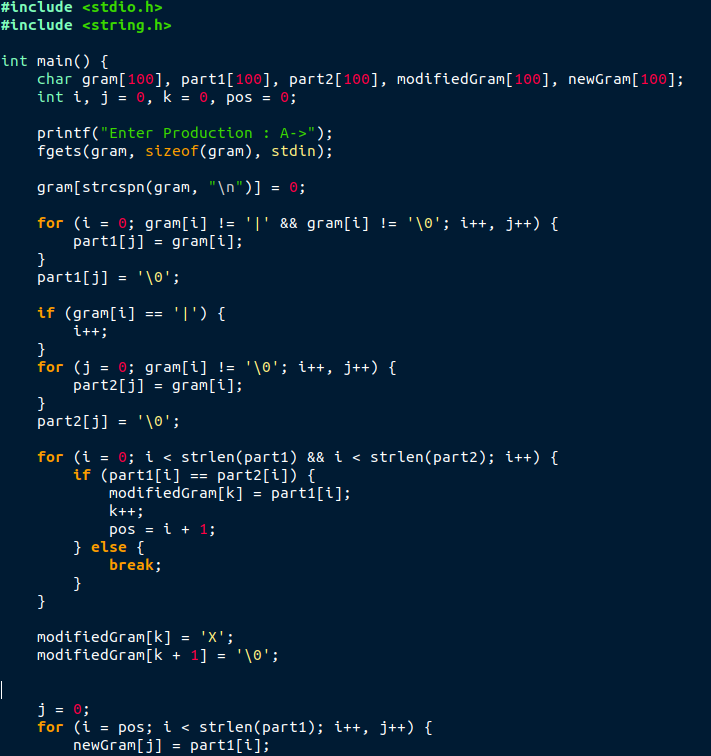
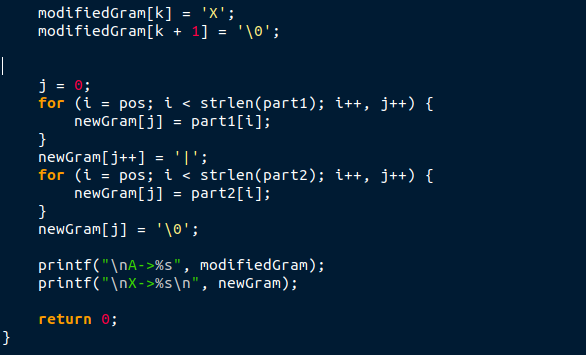
* Print each token with its appropriate label.
* End analysis.

CODE:  


RESULT:  


1. AIM: To program to eliminate left recursion and factoring from the given grammar.  
   ALGORITHM:

* Start the program.
* Input the grammar rules.
* For each non-terminal:
  1. Identify left-recursive and non-left-recursive productions.
  2. If left recursion exists:
     1. Eliminate using:
        1. A → βA'
        2. A' → αA' | ε
* For each non-terminal:
  1. Identify common prefixes.
  2. If found, apply left factoring:
     1. A → αA'
     2. A' → β1 | β2 | ...
* Display the updated grammar.
* End the program.

CODE:  
  
RESULT:  
