SSN COLLEGE OF ENGINEERING, KALAVAKKAM

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Compiler Design Lab – CS6612

Programming Assignment-3 - Implementation of Lexical Analyzer for the patterns (identifier, comments, operators, constants) using Lex

Due Date: 19.01.18 & 22.01.18

Develop a Lexical analyzer to recognize the patterns namely, identifiers, constants, comments and operators using the following regular expressions.

Regular Expression for Identifier	**Regular Expression for Constant**
letter → [a-zA-Z] digit → [0-9] id→letter(letter|digit)*	digit → [0-9] digits →digit digits optFrac →.digits optExp → E(+|-|ϵ) digits numberconst →digits optFrac optExp charconst → '(letter)' stringconst → "(letter)*" constant → numberconst | charconst |stringconst
Regular Expression for Comments	**Regular Expression for Operators**
start1→ \* end1 → */ multi → start (letter)* end start2 → // single → start (letter)*	relop → < | <= | == | != | > | >= arithop → + | - | * | / | % logicalop → && | || | ! operator → relop | arithop | logicalop

Regular Expression for keywords	
int → int	
float → float	
char → char	
double → double	
..	
..	
keywords → int|float|char|double|…..	

Convert the regular expressions into cumulative transition diagram as shown in Figure 1. Each state represents a condition that could occur during the process of scanning the input looking for a lexeme that matches one of the several patterns. Convert each state into a piece of code.

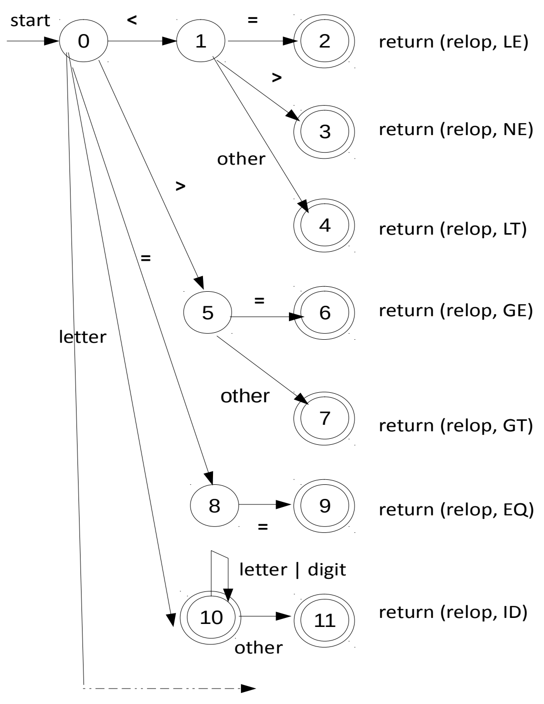

Figure 1. Cumulative Transition diagram

Develop a scanner that will recognize all the above specified tokens. Test your program for all specified tokens. Example input and output specification is given below.

EXAMPLE INPUT SOURCE PROGRAM

```
main()
{
  int a=10,b=20;
  if(a>b)
     printf("a is greater");
   else
     printf("b is greater");
}
```

OUTPUT

```
FC
SP
KW   ID   ASSIGN   NUMCONST   SP   ID   ASSIGN   NUMCONST   SP
KW SP ID RELOP SP
FC
KW
FC
SP
```