Exp2: Write a Lex Program to implement a Lexical Analyzer using Lex tool.

**Program: Lex1.l**

%{

int COMMENT=0;

%}

identifier [a-zA-Z][a-zA-Z0-9]\*

%%

#.\* {printf("\n%s is a preprocessor directive",yytext);}

int |

float |

char |

double |

while |

for |

struct |

typedef |

do |

if |

break |

continue |

void |

switch |

return |

else |

goto {printf("\n\t%s is a keyword",yytext);}

"/\*" {COMMENT=1;}{printf("\n\t %s is a COMMENT",yytext);}

{identifier}\( {if(!COMMENT)printf("\nFUNCTION \n\t%s",yytext);}

\{ {if(!COMMENT)printf("\n BLOCK BEGINS");}

\} {if(!COMMENT)printf("BLOCK ENDS ");}

{identifier}(\[[0-9]\*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}

\".\*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}

[0-9]+ {if(!COMMENT) printf("\n %s is a NUMBER ",yytext);}

\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}

\( ECHO;

= {if(!COMMENT)printf("\n\t %s is an ASSIGNMENT OPERATOR",yytext);}

\<= |

\>= |

\< |

== |

\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}

%%

int main(int argc, char \*\*argv)

{

FILE \*file;

file=fopen("var.c","r");

if(!file)

{ printf("could not open the file");

exit(0); }

yyin=file;

yylex();

printf("\n");

return(0);

}

int yywrap()

{ return(1); }

Program2 : var.c

#include<stdio.h>

void main()

{

int a,b,c;

a=1;

b=2;

c=a+b;

printf("Sum:%d",c);

}

**Execution:**

Step1: Run Program2

$vi var.c

$gcc var.c

$./a.out

Sum:3

Step2: Run Program 1

$vi lex1.l

$lex lex1.l

$gcc lex.yy.c

$./a.out

#include<stdio.h> is a preprocessor directive

void is a keyword

FUNCTION

main(

)

BLOCK BEGINS

int is a keyword

a IDENTIFIER,

b IDENTIFIER,

c IDENTIFIER;

a IDENTIFIER

= is an ASSIGNMENT OPERATOR

1 is a NUMBER ;

b IDENTIFIER

= is an ASSIGNMENT OPERATOR

2 is a NUMBER ;

c IDENTIFIER

= is an ASSIGNMENT OPERATOR

a IDENTIFIER+

b IDENTIFIER;

FUNCTION

printf(

"Sum:%d" is a STRING,

c IDENTIFIER

)

;

BLOCK ENDS