5: LEX Program

Aim: Design Lex program for to generate token of given input file.

Problem statement :-
Write a program using LEX specifications to implement lexical analysis phase of compiler to generate tokens of subset of Java program

Pre-requisities :- LEX 110, LEX 120, LEX 130, LEX 140, LEX 160, 250

Software requirements

S.No	Facilities req	Quantity
1.	System	1
2.	O/S	Ubuntu
3.	S/W name	LEX tool (flex)

Objectives :-

1) To understand LEX concepts
2) To implement LEX Program
3) To study about LEX & JAVA
4) To know important about Lexical analyzer.

Theory :-

Regular Expression in LEx

A regular expressions is a pattern description using a Meta Language. An expression is made up of symbols. Normal symbols are characters and numbers, but there are other symbols that have special meaning in LEX.

Programming in LEX :-

Programming in LEX can be divided into 3 steps :-
1) Specify the pattern - associated actions in a form
2) Run LEX over this file to generate C code for the
3) compile and link the C code to produce the executable scanner

```
... definitions ...
%  % rules
.... rules.....
%  %
.... subroutines...
```

Conclusion :-

Thus, we have studied lexical analyzer and implemented an applicn for lexical analyzer to perform scan the program and generates token of subset of java.

Assignment No. 05 [LEX Program]

Problem Satement: Write a program using Lex specifications to implement lexical analysis Phase of compiler to generate tokens of subset of Java program.

1. Code b2.l:

```
%{
   FILE* yyin;
%}

DATATYPE "int"|"char"|"float"|"double"
KEYWORDS "class"|"static"
DIGIT [0-9]
NUMBER {DIGIT}+
TEXT [a-zA-Z]
IDENTIFIER {TEXT}({DIGIT}|{TEXT}|"_")*
ACCESS "public"|"private"|"protected"
CONDITIONAL "if"|"else"|"else if"|"switch"
LOOP "for"|"while"|"do"
FUNCTION {ACCESS}{DATATYPE}{IDENTIFIER}"("({DATATYPE}{IDENTIFIER})*")"

%%
[  \n\t]+ ;
{ DATATYPE} {printf("%s == DATATYPE\n",yytext); }
{ KEYWORDS} {printf("%s == KEYWORDS\n",yytext); }
{ NUMBER} {printf("%s == NUMBER\n",yytext); }
{ IDENTIFIER} {printf("%s == IDENTIFIER\n",yytext); }

{ CONDITIONAL} {printf("%s == CONDITIONAL\n",yytext); }

{ FUNCTION} {printf("%s == FUNCTION\n",yytext); }
. ;
%%

int yywrap(){

}


int main(int argc,char* argv[]){
yyin= fopen(argv[1],"r");
   yylex();
fclose(yyin);
   return 0;
}
```

2. Demo.java Code:

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.Arrays;

public class demo
{


                public static void main(String[] args)throws Exception
                        { int hit=0; int miss=0;

                        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));

                        System.out.println("Enter total no of frames");
                        int noFrames=Integer.parseInt(br.readLine());

                        int[] frames=new int[noFrames];
                        int[] lruTime=new int[noFrames];

                        System.out.println("Enter total no of pages");
                        int totalPages = Integer.parseInt(br.readLine());

                        for(int i=0;i<totalPages;i++){
                                System.out.println("Enter page value");

                                int page= Integer.parseInt(br.readLine());

                                int searchIndex=isPresent(frames, page );


                                  if(searchIndex!=-1){
//      page fonud
                                                hit++; lruTime[searchIndex]=i;
                                                System.out.println("Page
                                                Hit");
                                }
                                else{
                                        System.out.println("Page Miss");
                                        miss++;


//      page not found
                                        int emptyindex=isEmpty(frames); if(emptyindex!=-
                                        1){
//      if frame is empty
                                                frames[emptyindex]=page;

                                                lruTime[emptyindex]=i;
                                        }
                                        else{
//user lru algo to find replace location
                                                int minLocationIndex=lru(lruTime);
```

```
                                                            System.out.println("Replace "+
frames[minLocationIndex]);

                                                            frames[minLocationIndex]=page;
                                                            lruTime[minLocationIndex]=i;

                                                        }

                                            }

                          }


                                      System.out.println("Total page hit" + hit);
                                  System.out.println("Total Page miss " + miss);
                                   System.out.println(Arrays.toString(frames));

                }

                public static int lru(int[] lruTime){ int min = 9999; int
                                            index = -1; for(int
                                            i=0;i<lruTime.length;i++){

                                                    if(min>lruTime[i]){
                                                            min=lruTime[i];
                                                            index=i;
                                                        }

                                            }

                                                  return index;
                }

                public static int isEmpty(int[] frames){

                                        for(int i=0;i<frames.length;i++)
                                { if(frames[i]==0){
                                            return i;
                                            }
                                }

                                return -1;
                }

                public static int isPresent(int[] frames, int search){

                          for(int i=0;i<frames.length;i++){
                                if(frames[i]==search)
                                                    return i;
                          }
```

```
                    return -1;
            }


}
```

OUTPUT:

```
Pritam-spos@Pritam-HP:~/SPOSL/LexProgram$ lex b2.l sagar-ravan@Sagar-
HP:~/SPOSL/LexProgram$ gcc lex.yy.c Pritam-spos@Pritam-HP:~/SPOSL/LexProgram$ ./a.out
demo.java
import == IDENTIFIER java ==
IDENTIFIER io ==
IDENTIFIER BufferedReader
== IDENTIFIER import ==
IDENTIFIER java ==
IDENTIFIER io ==
IDENTIFIER
InputStreamReader == IDENTIFIER
import == IDENTIFIER java ==
IDENTIFIER util == IDENTIFIER
Arrays == IDENTIFIER public ==
IDENTIFIER
class == KEYWORDS
demo == IDENTIFIER
public == IDENTIFIER
static == KEYWORDS
void == IDENTIFIER main
== IDENTIFIER String ==
IDENTIFIER args ==
IDENTIFIER throws ==
IDENTIFIER Exception ==
IDENTIFIER int ==
DATATYPE hit ==
IDENTIFIER 0  ==
NUMBER int ==
DATATYPE miss ==
IDENTIFIER 0  ==
NUMBER
BufferedReader == IDENTIFIER br
== IDENTIFIER new ==
IDENTIFIER BufferedReader ==
IDENTIFIER new == IDENTIFIER
InputStreamReader == IDENTIFIER
System == IDENTIFIER in ==
IDENTIFIER System ==
IDENTIFIER out == IDENTIFIER
```

```
println == IDENTIFIER Enter ==
IDENTIFIER total == IDENTIFIER
no == IDENTIFIER of ==
IDENTIFIER frames ==
IDENTIFIER int == DATATYPE
noFrames == IDENTIFIER Integer
== IDENTIFIER parseInt ==
IDENTIFIER br == IDENTIFIER
readLine == IDENTIFIER int ==
DATATYPE frames ==
IDENTIFIER new == IDENTIFIER
int == DATATYPE noFrames ==
IDENTIFIER int == DATATYPE
lruTime == IDENTIFIER new ==
IDENTIFIER int == DATATYPE
noFrames == IDENTIFIER System
== IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Enter ==
IDENTIFIER
total == IDENTIFIER no ==
IDENTIFIER of ==
IDENTIFIER pages ==
IDENTIFIER int ==
DATATYPE totalPages ==
IDENTIFIER Integer ==
IDENTIFIER parseInt ==
IDENTIFIER br ==
IDENTIFIER readLine ==
IDENTIFIER for ==
IDENTIFIER int ==
DATATYPE i ==
IDENTIFIER 0  ==
NUMBER i == IDENTIFIER
totalPages == IDENTIFIER i
== IDENTIFIER System ==
IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Enter ==
IDENTIFIER page ==
IDENTIFIER value ==
IDENTIFIER int ==
DATATYPE page ==
IDENTIFIER Integer ==
IDENTIFIER parseInt ==
IDENTIFIER br ==
IDENTIFIER readLine ==
IDENTIFIER int ==
DATATYPE searchIndex ==
IDENTIFIER isPresent ==
IDENTIFIER frames ==
IDENTIFIER page ==
IDENTIFIER if ==
IDENTIFIER searchIndex ==
IDENTIFIER 1  ==
NUMBER page ==
IDENTIFIER fonud ==
IDENTIFIER hit ==
IDENTIFIER lruTime ==
IDENTIFIER searchIndex ==
IDENTIFIER i ==
IDENTIFIER System ==
IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Page ==
IDENTIFIER Hit ==
IDENTIFIER else ==
IDENTIFIER System ==
IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER
Page == IDENTIFIER Miss ==
IDENTIFIER miss ==
IDENTIFIER page ==
IDENTIFIER not == IDENTIFIER
found == IDENTIFIER int ==
DATATYPE emptyindex ==
IDENTIFIER isEmpty ==
IDENTIFIER frames ==
IDENTIFIER if == IDENTIFIER
emptyindex == IDENTIFIER 1  ==
NUMBER if == IDENTIFIER
frame == IDENTIFIER is ==
IDENTIFIER empty ==
IDENTIFIER frames ==
IDENTIFIER emptyindex ==
IDENTIFIER page ==
IDENTIFIER lruTime ==
IDENTIFIER emptyindex ==
IDENTIFIER i == IDENTIFIER
else == IDENTIFIER user ==
IDENTIFIER lru == IDENTIFIER
algo == IDENTIFIER to ==
IDENTIFIER find == IDENTIFIER
replace == IDENTIFIER location
== IDENTIFIER int ==
DATATYPE minLocationIndex ==
IDENTIFIER lru == IDENTIFIER
lruTime == IDENTIFIER System
== IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Replace ==
IDENTIFIER frames ==
IDENTIFIER minLocationIndex ==
IDENTIFIER frames ==
IDENTIFIER minLocationIndex ==
IDENTIFIER page ==
IDENTIFIER lruTime ==
IDENTIFIER minLocationIndex ==
IDENTIFIER i == IDENTIFIER
System == IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Total ==
IDENTIFIER page ==
IDENTIFIER
hit == IDENTIFIER hit
== IDENTIFIER System
== IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Total ==
IDENTIFIER Page ==
IDENTIFIER miss ==
IDENTIFIER miss ==
IDENTIFIER System ==
IDENTIFIER out ==
IDENTIFIER println ==
IDENTIFIER Arrays ==
IDENTIFIER toString ==
IDENTIFIER frames ==
IDENTIFIER public ==
IDENTIFIER static ==
KEYWORDS int ==
DATATYPE lru ==
IDENTIFIER int ==
DATATYPE lruTime ==
IDENTIFIER int ==
DATATYPE min ==
IDENTIFIER 9999 ==
NUMBER int ==
DATATYPE index ==
IDENTIFIER 1  ==
NUMBER for ==
IDENTIFIER int ==
DATATYPE i ==
IDENTIFIER 0  ==
NUMBER i ==
IDENTIFIER lruTime ==
IDENTIFIER length ==
IDENTIFIER i ==
IDENTIFIER if ==
IDENTIFIER min ==
IDENTIFIER lruTime ==
IDENTIFIER i ==
IDENTIFIER min ==
IDENTIFIER lruTime ==
IDENTIFIER i ==
IDENTIFIER index ==
IDENTIFIER i ==
IDENTIFIER return ==
IDENTIFIER index ==
IDENTIFIER public ==
IDENTIFIER static ==
KEYWORDS int ==
DATATYPE isEmpty ==
IDENTIFIER int ==
DATATYPE frames ==
IDENTIFIER for ==
IDENTIFIER int ==
DATATYPE i ==
IDENTIFIER 0  ==
NUMBER i ==
IDENTIFIER frames ==
IDENTIFIER length ==
IDENTIFIER i ==
IDENTIFIER if ==
IDENTIFIER frames ==
IDENTIFIER i ==
IDENTIFIER 0  ==
NUMBER
return == IDENTIFIER
i == IDENTIFIER
return == IDENTIFIER
1  == NUMBER
public == IDENTIFIER
static == KEYWORDS int
== DATATYPE isPresent
== IDENTIFIER int ==
DATATYPE frames ==
IDENTIFIER int ==
DATATYPE search ==
IDENTIFIER for ==
IDENTIFIER int ==
DATATYPE i ==
IDENTIFIER 0  ==
NUMBER i ==
IDENTIFIER frames ==
IDENTIFIER length ==
IDENTIFIER i ==
IDENTIFIER if ==
IDENTIFIER frames ==
IDENTIFIER i ==
IDENTIFIER search ==
IDENTIFIER return ==
IDENTIFIER i ==
IDENTIFIER return ==
IDENTIFIER 1  ==
NUMBER
Pritam-spos@Pritam-HP:~/SPOSL/LexProgram$
```