DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

Year: 3rd **Semester: 6th**

Compiler Design - PCS-652

LAB MANUAL

Prepared By: HOD(CSE)

DEV BHOOMI INSTITUTE OF TECHNOLOGY

Department of Computer Science and Engineering

INDEX

S.No	Practical's Name	Date	Remark
1	Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines.		
2	Write a C program to identify whether a given line is a comment or not.		
3	Write a C program to recognize strings under 'a*', 'a*b+', 'abb'		
4	Write a C program to test whether a given identifier is valid or not		
5	Write a C program to simulate lexical analyzer for validating operators		
6	Implement the lexical analyzer using JLex, flex or other lexical analyzer generating tools.		

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 1**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value. Simulate the same in C language

RESOURCE: Turbo C ++

PROGRAM LOGIC:

1. Read the input Expression
2. Check whether input is alphabet or digits then store it as identifier
3. If the input is is operator store it as symbol
4. Check the input for keywords 1.4 PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program

Program:

```
#include<stdio.h>
 #include<conio.h>
 #include<string.h>
 char prol[7][10]={"S","A","A","B","B","C","C"}
 char pror[7][10]={"A","Bb","Cd","aB","@","Cc","@"}
 char prod[7][10]={"S->A","A->Bb","A->Cd","B->aB","B->@","C->Cc","C->@"}
 char first[7][10]={"abcd","ab","cd","a@","@","c@","@"}
 char follow[7][10]={"$","$","$","a$","b$","c$","d$"}
 char table[5][6][10]
 numr(char c)  {     switch(c)     {       case 'S': return 0
       case 'A': return 1
       case 'B': return 2
       case 'C': return 3
       case 'a': return 0
       case 'b': return 1
       case 'c': return 2
       case 'd': return 3
       case '$': return 4
     }
```

```
   return(2)
  } void main()  {     int i,j,k
    clrscr()
    for(i=0
i<5
i++)    for(j=0
j<6
j++)       strcpy(table[i][j]," ")
     printf("\nThe following is the predictive parsing table for the following grammar:\n")
    for(i=0
i<7
i++)       printf("%s\n",prod[i])
    printf("\nPredictive parsing table is\n")
    fflush(stdin)
     for(i=0
i<7
i++)     {       k=strlen(first[i])
       for(j=0
j<10
j++)        if(first[i][j]!='@')         strcpy(table[numr(prol[i][0])+1][numr(first[i][j])+1],prod[i])
     }    for(i=0
i<7
i++)     {       if(strlen(pror[i])==1)        {         if(pror[i][0]=='@')          {     k=strlen(follow[i])
     for(j=0
j<k
j++)
     strcpy(table[numr(prol[i][0])+1][numr(follow[i][j])+1],prod[i])

         }       }    }   strcpy(table[0][0]," ")
   strcpy(table[0][1],"a")
   strcpy(table[0][2],"b")
   strcpy(table[0][3],"c")
   strcpy(table[0][4],"d")
   strcpy(table[0][5],"$")
   strcpy(table[1][0],"S")
   strcpy(table[2][0],"A")
   strcpy(table[3][0],"B")
   strcpy(table[4][0],"C")
   printf("\n-------------------------------------------------------\n")
   for(i=0
i<5
i++)    for(j=0
j<6
j++)      {       printf("%-10s",table[i][j])
```

```
      if(j==5)        printf("\n-------------------------------------------------------\n")
     }   getch()
  }
```

PRE LAB QUESTIONS

1. What is token?
2. What is lexeme?
3. What is the difference between token and lexeme?
4. Define phase and pass?
5. What is the difference between phase and pass?
6. What is the difference between compiler and interpreter?

LAB ASSIGNMENT

1. Write a program to recognize identifiers.
2. Write a program to recognize constants.
3. Write a program to recognize keywords and identifiers.
4. Write a program to ignore the comments in the given input source program.

POST LAB QUESTIONS

1. What is lexical analyzer?
2. Which compiler is used for lexical analyzer?
3. What is the output of Lexical analyzer?
4. What is LEX source Program?

INPUT & OUTPUT:

Input: Enter Program $ for termination:

```
 {
   int a[3],t1,t2
   t1=2
a[0]=1
 a[1]=2
 a[t1]=3
 t2=-(a[2]+t1*6)/(a[2]-t1)
     if t2>5 then      print(t2)
    else {    int t3
      t3=99
     t2=-25
     print(-t1+t2*t3)
   /* this is a comment   on 2 lines */   }  endif  }
```

$ Output: Variables : a[3] t1 t2 t3 Operator : - + * / > Constants : 2 1 3 6 5 99 -25 Keywords : int if then else endif Special Symbols : , ( ) { } Comments : this is a comment on 2 lines

```
  c=getc(f1)
   while(isdigit(c))     {       tokenvalue*=10+c-'0'
      c=getc(f1)
     }           num[i++]=tokenvalue
          ungetc(c,f1)
 } else                if(isalpha(c))                    {    putc(c,f2)
   c=getc(f1)
   while(isdigit(c)||isalpha(c)||c=='_'||c=='$')        {     putc(c,f2)
    c=getc(f1)
       }    putc(' ',f2)
   ungetc(c,f1)
  }        else            if(c==' '||c=='\t')    printf(" ")
        else    if(c=='\n')     lineno++
   else     putc(c,f3)
            } fclose(f2)
 fclose(f3)
 fclose(f1)
 printf("\nThe no's in the program are")
 for(j=0
j<i
j++) printf("%d",num[j])
 printf("\n")
 f2=fopen("identifier","r")
 k=0
 printf("The keywords and identifiersare:")
 while((c=getc(f2))!=EOF)         {     if(c!=' ')    str[k++]=c
   else        {            str[k]='\0'
                     keyword(str)
          k=0
     }
                    }    fclose(f2)
 f3=fopen("specialchar","r")
  printf("\nSpecial characters are")
  while((c=getc(f3))!=EOF)   printf("%c",c)
  printf("\n")
  fclose(f3)
  printf("Total no. of lines are:%d",lineno)
 }

}
```

PRE LAB QUESTIONS

1. What is token?
2. What is lexeme?
3. What is the difference between token and lexeme?

4. Define phase and pass?
5. What is the difference between phase and pass?
6. What is the difference between compiler and interpreter?

LAB ASSIGNMENT

1. Write a program to recognize identifiers.
2. Write a program to recognize constants.
3. Write a program to recognize keywords and identifiers.
4. Write a program to ignore the comments in the given input source program.

POST LAB QUESTIONS

1. What is lexical analyzer?
2. Which compiler is used for lexical analyzer?
3. What is the output of Lexical analyzer?
4. What is LEX source Program?

INPUT & OUTPUT:

Input: Enter Program $ for termination: { int a[3],t1,t2
t1=2
a[0]=1
a[1]=2
a[t1]=3
t2=-(a[2]+t1*6)/(a[2]-t1)
if t2>5 then print(t2)
else { int t3
t3=99
t2=-25
print(-t1+t2*t3)
/* this is a comment on 2 lines */ } endif }
$ **Output:** Variables : a[3] t1 t2 t3 Operator : - + * / > Constants : 2 1 3 6 5 99 -25 Keywords : int if then else endif Special Symbols : , ( ) { } Comments : this is a comment on 2 lines

OUTCOMES: Students will understand the lexical analyzer for a given using C.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 2**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Write a C program to identify whether a given line is a comment or not.
RESOURCE: Turbo C++
PROGRAM LOGIC:
Read the input string. Check whether the string is starting with '/' and check next character is '/' or'*'. If condition satisfies print comment. Else not a comment.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.
PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
    {  char com[30]
 int i=2,a=0
 clrscr()
 printf("\n Enter comment:")
 gets(com)
 if(com[0]=='/')  {      if(com[1]=='/')        printf("\n It is a comment")
     else if(com[1]=='*')   {       for(i=2
i<=30
i++)       {         if(com[i]=='*'&&com[i+1]=='/')          {          printf("\n It is a comment")
            a=1
            break
   }         else           continue
 }       if(a==0)         printf("\n It is not a comment")
    }     else     printf("\n It is not a comment")
  }  else     printf("\n It is not a comment")
 getch()
 }
```

INPUT & OUTPUT:
Input: Enter comment: //hello
Output: It is a comment
Input: Enter comment: hello
Output: It is not a comment

OUTCOMES: Students will implement and identify whether a given line is a comment or not.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 3**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Write a C program to recognize strings under 'a*', 'a*b+', 'abb'.
RESOURCE: Turbo C++
PROGRAM LOGIC: By using transition diagram we verify input of the state. If the state recognize the given pattern rule. Then print string is accepted under a*/ a*b+/ abb. Else print string not accepted.
PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.
PROGRAM:

```
 #include<stdio.h>
#include<conio.h>
#include<string.h>
 #include<stdlib.h>
void main()
 {    char s[20],c
  int state=0,i=0
   clrscr()
  printf("\n Enter a string:")
 gets(s)
  while(s[i]!='\0')     {       switch(state)        {          case 0: c=s[i++]
                    if(c=='a')                          state=1
                    else if(c=='b')                       state=2
                              else           state=6
             break
        case 1: c=s[i++]
          if(c=='a')              state=3

          else if(c=='b')           state=4
       else           state=6
          break
        case 2: c=s[i++]
          if(c=='a')              state=6
          else if(c=='b')           state=2
       else           state=6
          break
         case 3: c=s[i++]
          if(c=='a')              state=3
```

```
        else if(c=='b')          state=2
      else          state=6
        break
       case 4: c=s[i++]
        if(c=='a')              state=6
                    else if(c=='b')          state=5
      else          state=6
        break
       case 5: c=s[i++]
        if(c=='a')              state=6
        else if(c=='b')          state=2
      else          state=6
         break
       case 6: printf("\n %s is not recognised.",s)
        exit(0)
      }     }
  I f(state==1)       printf("\n %s is accepted under rule 'a'",s)
   else if((state==2)||(state==4))     printf("\n %s is accepted under rule 'a*b+'",s)
    else if(state==5)         printf("\n %s is accepted under rule 'abb'",s)
   getch()
 }
```

INPUT & OUTPUT:

Input : Enter a String: aaaabbbbb

Output: aaaabbbbb is accepted under rule 'a*b+'

Enter a string: cdgs cdgs is not recognized

OUTCOMES: Students will recognize strings under 'a*', 'a*b+', 'abb'.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 4**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Write a C program to test whether a given identifier is valid or not.

RESOURCE: Turbo C++

PROGRAM LOGIC: Read the given input string. Check the initial character of the string is numerical or any special character except '_' then print it is not a valid identifier. Otherwise print it as valid identifier if remaining characters of string doesn't contains any special characters except '_'.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
 {
   char a[10]
  int flag, i=1
  clrscr()
  printf("\n Enter an identifier:")
  gets(a)
   if(isalpha(a[0]))      flag=1
  else      printf("\n Not a valid identifier")
   while(a[i]!='\0')    {   if(!isdigit(a[i])&&!isalpha(a[i]))       {         flag=0
        break
      }     i++
  }  if(flag==1)     printf("\n Valid identifier")
   getch()
 }
```

INPUT & OUTPUT:

Input:

Enter an identifier: first

Output:

Valid identifier

Enter an identifier:1aqw

Not a valid identifier

OUTCOMES: Students will test whether a given identifier is valid or not.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 5**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Write a C program to simulate lexical analyzer for validating operators.

RESOURCE: Turbo C++

PROGRAM LOGIC : Read the given input. If the given input matches with any operator symbol. Then display in terms of words of the particular symbol. Else print not a operator.

PROCEDURE: Go to debug -> run or press CTRL + F9 to run the program.

PROGRAM:

```
#include<stdio.h>
 #include<conio.h>
 void main()
 {
   char s[5]
   clrscr()
    printf("\n Enter any operator:")
    gets(s)
    switch(s[0])    {    case'>': if(s[1]=='=')          printf("\n Greater than or equal")
            else           printf("\n Greater than")
            break
   case'<': if(s[1]=='=')          printf("\n Less than or equal")
          else          printf("\nLess than")
          break
   case'=': if(s[1]=='=')          printf("\nEqual to")
          else          printf("\nAssignment")
          break
   case'!': if(s[1]=='=')          printf("\nNot Equal")
          else          printf("\n Bit Not")
          break
   case'&': if(s[1]=='&')          printf("\nLogical AND")
          else          printf("\n Bitwise AND")
          break
   case'|': if(s[1]=='|')          printf("\nLogical OR")

         else          printf("\nBitwise OR")
          break
   case'+': printf("\n Addition")
```

```
        break
 case'-': printf("\nSubstraction")
        break
 case'*': printf("\nMultiplication")
        break
case'/': printf("\nDivision")
       break
case'%': printf("Modulus")
       break
default: printf("\n Not a operator")
 }   getch()
}
```

INPUT & OUTPUT:

Input Enter any operator: *

Output Multiplication

OUTCOMES: Students will simulate lexical analyzer for validating operators.

DEV BHOOMI INSTITUTE OF TECHNOLOGY

LAB MANUAL

	Course Name : Compiler Design	**EXPERIMENT NO. 6**	
	Course Code : PCS 652 **Faculty : Mr. Shubhashish Goswami**	**Branch: CSE**	**Semester: VI**

OBJECTIVE: Implement the lexical analyzer using JLex, flex or other lexical analyzer generating tools.

RESOURCE: Linux using Putty

PROGRAM LOGIC: Read the input string. Check whether the string is identifier/ keyword /symbol by using the rules of identifier and keywords using LEX Tool

PROCEDURE: Go to terminal .Open vi editor ,Lex lex.l , cc lex.yy.c , ./a.out

PROGRAM:

```
/* program name is lexp.l */ %{ /* program to recognize a c program */ 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 |do |if |break |continue |void |switch |case |long |struct
|const |typedef |return |else |goto {printf("\n\t%s is a KEYWORD",yytext)
} "/*" {COMMENT = 1
} /*{printf("\n\n\t%s is a COMMENT\n",yytext)
}*/ "*/" {COMMENT = 0
} /* printf("\n\n\t%s is a COMMENT\n",yytext)
}*/ {identifier}\( {if(!COMMENT)printf("\n\nFUNCTION\n\t%s",yytext)
} { {if(!COMMENT) printf("\n BLOCK BEGINS")
} } {if(!COMMENT) printf("\n 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\t%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) { if (argc > 1) { FILE *file
 file = fopen(argv[1],"r")
 if(!file) { printf("could not open %s \n",argv[1])
 exit(0)
 } yyin = file
 } yylex()
```

```
printf("\n\n")
 return 0
 } int yywrap() { return 0
 }
```

PRE LAB QUESTIONS:

1. List the different sections available in LEX compiler?
2. What is an auxiliary definition?
3. How can we define the translation rules?
4. What is regular expression?
5. What is finite automaton?

LAB ASSIGNMENT:

1. Write a program that defines auxiliary definitions and translation rules of Pascal tokens?
2. Write a program that defines auxiliary definitions and translation rules of C tokens?
3. Write a program that defines auxiliary definitions and translation rules of JAVA tokens

POST LAB QUESTIONS:

1. What is Jlex?
2. What is Flex?
3. What is lexical analyzer generator?
4. What is the input for LEX Compiler?
5. What is the output of LEX compiler?

INPUT & OUTPUT:

Input

```
$vi var.c #include<stdio.h> main() { int a,b
 }
```

Output

$lex lex.l $cc lex.yy.c $./a.out var.c #include<stdio.h> is a PREPROCESSOR DIRECTIVE FUNCTION main ( ) BLOCK BEGINS int is a KEYWORD a IDENTIFIER b IDENTIFIER BLOCK ENDS

OUTCOMES: Students will Implement the lexical analyzer using JLex, flex and generating tools.