**Name: Atharva Paliwal**

**Roll No: 6B-40**

**Practical No. 1**

**Theory**

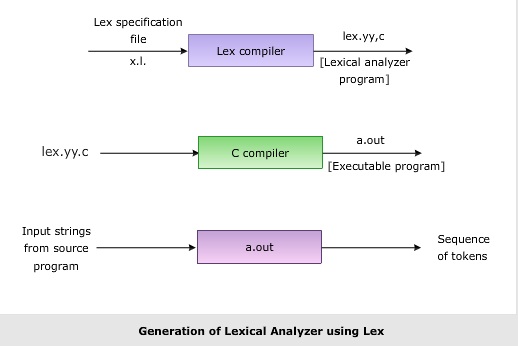
**LEX:**

Lex is a program generator designed for lexical processing of character input streams. It accepts a high level, problem-oriented specification for character string matching, and produces a program in a general-purpose language which recognizes regular expressions. The regular expressions are specified by the user in the source specifications given to Lex. The Lex written code recognizes these expressions in an input stream and partitions the input stream into strings matching the expressions. At the boundaries between strings program sections provided by the user are executed. The Lex source file associates the regular expressions and the program fragments. As each expression appears in the input to the program written by Lex, the corresponding fragment is executed.

Lex is not a complete language, but rather a generator representing a new language feature which can be added to different programming languages, called ``host languages.'' Just as general purpose languages can produce code to run on different computer hardware, Lex can write code in different host languages.

Lex turns the user's expressions and actions (called source in this pic) into the host general-purpose language; the generated program is named yylex. The yylex program will recognize expressions in a stream (called input in this pic) and perform the specified actions for each expression as it is detected.

**Diagram of LEX**



**Format for Lex file**

The general format of Lex source is:

{definitions}

%%

{rules}

%%

{user subroutines}

where the definitions and the user subroutines are often omitted. The second %% is optional, but the first is required to mark the beginning of the rules. The absolute minimum Lex program is thus %% (no definitions, no rules) which translates into a program which copies the input to the output unchanged.

**Regular Expression**

A regular expression (or RE) specifies a set of strings that matches it; the functions in this module let you check if a particular string matches a given regular expression (or if a given regular expression matches a particular string, which comes down to the same thing).

Regular expressions can be concatenated to form new regular expressions; if A and B are both regular expressions, then AB is also a regular expression. In general, if a string p matches A and another string q matches B, the string pqwill match AB. This holds unless A or B contain low precedence operations; boundary conditions between A and B; or have numbered group references. Thus, complex expressions can easily be constructed from simpler primitive expressions. Regular expressions can contain both special and ordinary characters. Most ordinary characters, like "A", "a", or "0", are the simplest regular expressions; they simply match themselves. You can concatenate ordinary characters, so last matches the string 'last'. (In the rest of this section, we'll write RE's in this special style, usually without quotes, and strings to be matched 'in single quotes'.)

Some characters, like "|" or "(", are special. Special characters either stand for classes of ordinary characters or affect how the regular expressions around them are interpreted.

**Lex Library Routines**

Lex library routines are those functions which have a detailed knowledge of the lex functionalities and which can be called to implement various tasks in a lex program.

The following table gives a list of some of the lex routines.

|  |  |
| --- | --- |
| Lex Routine | Description |
| Main () | Invokes the lexical analyzer by calling the yylex subroutine. |
| yywrap () | Returns the value 1 when the end of input occurs. |
| yymore () | Appends the next matched string to the current value of the yytext array rather than replacing the contents of the yytext array. |
| yyless (int n) | Retains n initial characters in the yytext array and returns the remaining characters to the input stream. |
| yyreject | Allows the lexical analyzer to match multiple rules for the same input string. (The yyreject subroutine is called when the special action REJECT is used.) |
| yylex () | The default main () contains the call of yylex () |

**Answer the Questions:**

1. Why is –ll option used for running lex.yy.c?

The file *lex.yy.c* may be compiled and linked in the same way as any C program. The **-ll** option is used to link the object file created from this C source with **lex** library: **cc lex.yy.c -ll**

1. Use of yywrap

Function **yywrap** is called by lex when input is exhausted. Return 1 if you are done or 0 if more processing is required. Every C program requires a main function. In this case we simply call yylex that is the main entry-point for lex.

1. Use of yylex function

Implies the main entry point for lex, reads the input stream generates tokens, returns zero at the end of input stream. It is called to invoke the lexer (or scanner) and each time yylex () is called, the scanner continues processing the input from where it last left off.

1. What does lex.yy.c. do?

The **lex** command stores the yylex function in a file named **lex**. **yy**.c You **can** use the yylex function alone to recognize simple one-word input, or you **can** use it with other **C** language programs to perform more difficult input analysis functions.

**Practical No. 1 (A)**

**Aim:** Design a lexical analyzer to identify the tokens such as keywords, identifiers, operators, constants, symbols and strings for C language using LEX.

**Program:**

%{

#include<stdio.h>

int functions=0, logicaloperators=0, assignmentoperators=0, arithmeticoperators=0 ,relationaloperators=0, keywords=0, integerconstants=0, floatingconstants=0, charconstants=0, identifiers=0, symbol=0, strings=0, lines=0;

%}

%%

"#include".\* {printf("\n%s is HEADER\n",yytext);}

scanf|printf {printf("\n%s is FUNCTION\n",yytext); functions++;}

[&&]|[!]|[||] {printf("\n%s is LOGICAL OPERATOR\n",yytext);logicaloperators++;}

[-+%/\*] {printf("\n%s is ARITHMETIC OPERATOR\n",yytext);arithmeticoperators++;}

[=]|[-=]|[+=] {printf("\n%s is ASSIGNMENT OPERATOR\n",yytext);assignmentoperators++;}

[==]|[!=]|[<=]|[>=]|[<>] {printf("\n%s is RELATIONAL OPERATOR\n",yytext);relationaloperators++;}

int|float|void|main|char|if|while {printf("\n%s is KEYWORD\n",yytext); keywords++;}

[0-9]+ {printf("\n%s is INTEGER CONSTANT\n ",yytext);integerconstants++;}

[0-9]+[.][0-9]+ {printf("\n%s is FLOATING CONSTANT\n ",yytext);floatingconstants++;}

['][a-zA-z]+['] {printf("\n%s is CHAR CONSTANT\n",yytext);charconstants++;}

[A-Za-z][A-Za-z0-9]\* {printf("\n%s is IDENTIFIER\n",yytext);identifiers++;}

[{}#()@$:;,?] {printf("\n%s is SPECIAL SYMBOL",yytext);symbol++;}

\".\*\" {printf("\n%s is STRING\n",yytext);strings++;}

\n lines++;

%%

int main(void)

{

yyin = fopen("expt1a.txt","r");

yylex();

printf("\n----------------------------------------------");

printf("\n%d KEYWORDS IN C FILE", keywords);

printf("\n%d FUNCTIONS IN C FILE", functions);

printf("\n%d ASSIGNMENT OPERATORS IN C FILE", assignmentoperators);

printf("\n%d RELATIONAL OPERATORS IN C FILE", relationaloperators);

printf("\n%d LOGICAL OPERATORS IN C FILE", logicaloperators);

printf("\n%d ARITHMETIC OPERATORS IN C FILE", arithmeticoperators);

printf("\n%d INTEGER CONSTANTS IN C FILE", integerconstants);

printf("\n%d CHAR CONSTANTS IN C FILE", charconstants);

printf("\n%d FLOATING CONSTANTS IN C FILE", floatingconstants);

printf("\n%d IDENTIFIERS IN C FILE", identifiers);

printf("\n%d STRINGS IN C FILE", strings);

printf("\n%d SYMBOLS IN C FILE", symbol);

printf("\n----------------------------------------------");

printf("\n%d LINES IN C FILE", lines);

}

int yywrap(){

return(1);

}

**Input:**

#include <stdio.h>

int main() {

int b = 10.0;

char ch='c' ;

int t1 = 0, t2 = 1, nextTerm = 0, n;

printf("Enter a positive number: ");

scanf("%d", &n);

printf("Fibonacci Series: %d, %d, ", t1, t2);

nextTerm = t1 + t2;

while (nextTerm <= n) {

printf("%d, ", nextTerm);

t1 = t2;

t2 = nextTerm;

nextTerm = t1 + t2;

}

return 0;

}

**Output:**

#include <stdio.h> is HEADER

int is KEYWORD

main is KEYWORD

( is SPECIAL SYMBOL

) is SPECIAL SYMBOL

{ is SPECIAL SYMBOL

int is KEYWORD

b is IDENTIFIER

= is ASSIGNMENT OPERATOR

10.0 is FLOATING CONSTANT

; is SPECIAL SYMBOL

char is KEYWORD

ch is IDENTIFIER

= is ASSIGNMENT OPERATOR

'c' is CHAR CONSTANT

; is SPECIAL SYMBOL

int is KEYWORD

t1 is IDENTIFIER

= is ASSIGNMENT OPERATOR

0 is INTEGER CONSTANT

, is SPECIAL SYMBOL

t2 is IDENTIFIER

= is ASSIGNMENT OPERATOR

1 is INTEGER CONSTANT

, is SPECIAL SYMBOL

nextTerm is IDENTIFIER

= is ASSIGNMENT OPERATOR

0 is INTEGER CONSTANT

, is SPECIAL SYMBOL

n is IDENTIFIER

; is SPECIAL SYMBOL

printf is FUNCTION

( is SPECIAL SYMBOL

"Enter a positive number: " is STRING

) is SPECIAL SYMBOL

; is SPECIAL SYMBOL

scanf is FUNCTION

( is SPECIAL SYMBOL

"%d" is STRING

, is SPECIAL SYMBOL

& is LOGICAL OPERATOR

n is IDENTIFIER

) is SPECIAL SYMBOL

; is SPECIAL SYMBOL

printf is FUNCTION

( is SPECIAL SYMBOL

"Fibonacci Series: %d, %d, " is STRING

, is SPECIAL SYMBOL

t1 is IDENTIFIER

, is SPECIAL SYMBOL

t2 is IDENTIFIER

) is SPECIAL SYMBOL

; is SPECIAL SYMBOL

nextTerm is IDENTIFIER

= is ASSIGNMENT OPERATOR

t1 is IDENTIFIER

+ is ARITHMETIC OPERATOR

t2 is IDENTIFIER

; is SPECIAL SYMBOL

while is KEYWORD

( is SPECIAL SYMBOL

nextTerm is IDENTIFIER

< is RELATIONAL OPERATOR

= is ASSIGNMENT OPERATOR

n is IDENTIFIER

) is SPECIAL SYMBOL

{ is SPECIAL SYMBOL

printf is FUNCTION

( is SPECIAL SYMBOL

"%d, " is STRING

, is SPECIAL SYMBOL

nextTerm is IDENTIFIER

) is SPECIAL SYMBOL

; is SPECIAL SYMBOL

t1 is IDENTIFIER

= is ASSIGNMENT OPERATOR

t2 is IDENTIFIER

; is SPECIAL SYMBOL

t2 is IDENTIFIER

= is ASSIGNMENT OPERATOR

nextTerm is IDENTIFIER

; is SPECIAL SYMBOL

nextTerm is IDENTIFIER

= is ASSIGNMENT OPERATOR

t1 is IDENTIFIER

+ is ARITHMETIC OPERATOR

t2 is IDENTIFIER

; is SPECIAL SYMBOL

} is SPECIAL SYMBOL

return is IDENTIFIER

0 is INTEGER CONSTANT

; is SPECIAL SYMBOL

} is SPECIAL SYMBOL

----------------------------------------------

6 KEYWORDS IN C FILE

4 FUNCTIONS IN C FILE

10 ASSIGNMENT OPERATORS IN C FILE

1 RELATIONAL OPERATORS IN C FILE

1 LOGICAL OPERATORS IN C FILE

2 ARITHMETIC OPERATORS IN C FILE

4 INTEGER CONSTANTS IN C FILE

1 CHAR CONSTANTS IN C FILE

1 FLOATING CONSTANTS IN C FILE

23 IDENTIFIERS IN C FILE

4 STRINGS IN C FILE

35 SYMBOLS IN C FILE

----------------------------------------------

18 LINES IN C FILE

**Practical No. 1 (B)**

**Aim:** Create a txt file to containing the following without heading: Name of Student, Company

Placed in (TCS, Infosys, Wipro, Accenture, Informatica), Male/female, CGPA (floating point

number), Department (CSE, IT, EC), Package (floating point number), mail id, mobile number

(integer exactly 10 digits). At least 25 records must be present.

**Program:**

%{

    #include<stdio.h>

    #include<string.h>

    int male=0,female=0,tcs=0,infsys=0,wipro=0,acc=0,infmat=0,n=0,it=0,ec=0,cs=0;

%}

%%

[a-z.0-9]+@[a-z]+.[a-z]+ {n++;printf("\nEMAIL : %s", yytext);}

Male|Female {printf("\nGENDER : %s", yytext);

    if(strcmp(yytext,"Male")==0) male++;

    else if(strcmp(yytext,"Female")==0) female++;

}

CSE|EC|IT {printf("\nBRANCH : %s", yytext);

    if(strcmp(yytext,"CSE")==0) cs++;

    else if(strcmp(yytext,"IT")==0) it++;

    else if(strcmp(yytext,"EC")==0) ec++;

}

TCS|Infosys|Wipro|Accenture|Informatica {printf("\nCOMPANY : %s", yytext);

    if(strcmp(yytext,"TCS")==0) tcs++;

    else if(strcmp(yytext,"Infosys")==0) infsys++;

    else if(strcmp(yytext,"Wipro")==0) wipro++;

    else if(strcmp(yytext,"Accenture")==0) acc++;

    else if(strcmp(yytext,"Informatica")==0) infmat++;

}

[0-9][.][0-9] {n++;printf("\nCGPA : %s", yytext);}

[0-9]\*+"."+[0-9] {n++;printf("\nPACKAGE : %s", yytext);}

[8-9]{1}[0-9]{9} {n++;printf("\nMOBILE NO. : %s", yytext);}

[a-zA-Z\_]+ {n++;printf("\nNAME : %s", yytext);}

. ;

%%

int main(void)

{

    yyin=fopen("data.txt","r");

    yylex();

    printf("\n");

    printf("\n==============PLACEMENT STATISTICS===================\n");

    printf("\nSTUDENTS PLACED IN TCS = %d\n",tcs);

    printf("\nSTUDENTS PLACED IN INFOSYS = %d\n",infsys);

    printf("\nSTUDENTS PLACED IN WIPRO = %d\n",wipro);

    printf("\nSTUDENTS PLACED IN ACCENTURE = %d\n",acc);

    printf("\nSTUDENTS PLACED IN INFORMATICA = %d\n",infmat);

    printf("\n================STUDENTS STATISTICS=====================\n");

    printf("\nFEMALE STUDENTS PLACED = %d\n",female);

    printf("\nMALE STUDENTS PLACED = %d\n",male);

    printf("\n================BRANCH STATISTICS=====================\n");

    printf("\nSTUDENTS PLACED FROM CSE = %d\n",cs);

    printf("\nSTUDENTS PLACED FROM IT = %d\n",it);

    printf("\nSTUDENTS PLACED FROM EC = %d\n",ec);

}

int yywrap(void){

    return(1);

}

**Input:**

Atharva Informatica Male 9.1 CSE 1400000.0 paliwalap@rknec.edu 9809101421

Rishikesh Informatica Male 9.4 CSE 60000.0 kale@rknec.edu 9851407159

Rohit Infosys Male 8.4 EC 800000.0 chandani@rknec.edu 952502339252

Rahul Wipro Male 9.6 CSE 700000.0 agrawal@rknec.edu 083 8835605442

Riya Infosys Female 8.7 IT 1400000.0 riya@rknec.edu 9893674960

Monica TCS Female 9.9 IT 300000.0 monica@rknec.edu 9899625904

Anjali TCS Male 9.7 CSE 700000.0 rajendran@rknec.edu 8879917599

Simran Infosys Female 9.2 CSE 600000.0 baheti@rknec.edu 8289563742

Ankita Wipro TCS Female 8.9 EC 600000.0 ankita@rknec.edu 8356145169

Ayushi Infosys Female 9.4 CSE 200000.0 ayushi@rknec.edu 9328571185

Riddhi Informatica Female 6.4 CSE 900000.0 lele@rknec.edu 9028422193

Muskan Infosys Female 8.5 EC 1700000.0 gupta@rknec.edu 8432104405

Keshubh Wipro Male 9.8 IT 400000.0 sharma@rknec.edu 9060213371

Gaurav Infosys Male 9.4 CSE 200000.0 shukla@rknec.edu 9767840563

Ritik Accenture Male 7.1 EC 800000.0 parashar@rknec.edu 1009520064

Yaman Accenture Male 9.9 IT 1700000.0 kushwaha@rknec.edu 9861889583

Nikhil Wipro Male 9.2 CSE 900000.0 tiwari@rknec.edu 8671206472

Yash Informatica Male 8.4 EC 1400000.0 roy@rknec.edu 9692086015

Paritosh Wipro Male 9.5 CSE 100000.0 dandekar@rknec.edu 8954205715

**Output:**

NAME : Atharva

COMPANY : Informatica

GENDER : Male

CGPA : 9.1

BRANCH : CSE

PACKAGE : 1400000.0

EMAIL : paliwalap@rknec.edu

MOBILE NO. : 9809101421

NAME : Rishikesh

COMPANY : Informatica

GENDER : Male

CGPA : 9.4

BRANCH : CSE

PACKAGE : 60000.0

EMAIL : kale@rknec.edu

MOBILE NO. : 9851407159

NAME : Rohit

COMPANY : Infosys

GENDER : Male

CGPA : 8.4

BRANCH : EC

PACKAGE : 800000.0

EMAIL : chandani@rknec.edu

MOBILE NO. : 9525023392

NAME : Rahul

COMPANY : Wipro

GENDER : Male

CGPA : 9.6

BRANCH : CSE

PACKAGE : 700000.0

EMAIL : agrawal@rknec.edu

MOBILE NO. : 8835605442

NAME : Riya

COMPANY : Infosys

GENDER : Female

CGPA : 8.7

BRANCH : IT

PACKAGE : 1400000.0

EMAIL : riya@rknec.edu

MOBILE NO. : 9893674960

NAME : Monica

COMPANY : TCS

GENDER : Female

CGPA : 9.9

BRANCH : IT

PACKAGE : 300000.0

EMAIL : monica@rknec.edu

MOBILE NO. : 9899625904

NAME : Anjali

COMPANY : TCS

GENDER : Male

CGPA : 9.7

BRANCH : CSE

PACKAGE : 700000.0

EMAIL : rajendran@rknec.edu

MOBILE NO. : 8879917599

NAME : Simran

COMPANY : Infosys

GENDER : Female

CGPA : 9.2

BRANCH : CSE

PACKAGE : 600000.0

EMAIL : baheti@rknec.edu

MOBILE NO. : 8289563742

NAME : Ankita

COMPANY : Wipro

COMPANY : TCS

GENDER : Female

CGPA : 8.9

BRANCH : EC

PACKAGE : 600000.0

EMAIL : ankita@rknec.edu

MOBILE NO. : 8356145169

NAME : Ayushi

COMPANY : Infosys

GENDER : Female

CGPA : 9.4

BRANCH : CSE

PACKAGE : 200000.0

EMAIL : ayushi@rknec.edu

MOBILE NO. : 9328571185

NAME : Riddhi

COMPANY : Informatica

GENDER : Female

CGPA : 6.4

BRANCH : CSE

PACKAGE : 900000.0

EMAIL : lele@rknec.edu

MOBILE NO. : 9028422193

NAME : Muskan

COMPANY : Infosys

GENDER : Female

CGPA : 8.5

BRANCH : EC

PACKAGE : 1700000.0

EMAIL : gupta@rknec.edu

MOBILE NO. : 8432104405

NAME : Keshubh

COMPANY : Wipro

GENDER : Male

CGPA : 9.8

BRANCH : IT

PACKAGE : 400000.0

EMAIL : sharma@rknec.edu

MOBILE NO. : 9060213371

NAME : Gaurav

COMPANY : Infosys

GENDER : Male

CGPA : 9.4

BRANCH : CSE

PACKAGE : 200000.0

EMAIL : shukla@rknec.edu

MOBILE NO. : 9767840563

NAME : Ritik

COMPANY : Accenture

GENDER : Male

CGPA : 7.1

BRANCH : EC

PACKAGE : 800000.0

EMAIL : parashar@rknec.edu

NAME : Yaman

COMPANY : Accenture

GENDER : Male

CGPA : 9.9

BRANCH : IT

PACKAGE : 1700000.0

EMAIL : kushwaha@rknec.edu

MOBILE NO. : 9861889583

NAME : Nikhil

COMPANY : Wipro

GENDER : Male

CGPA : 9.2

BRANCH : CSE

PACKAGE : 900000.0

EMAIL : tiwari@rknec.edu

MOBILE NO. : 8671206472

NAME : Yash

COMPANY : Informatica

GENDER : Male

CGPA : 8.4

BRANCH : EC

PACKAGE : 1400000.0

EMAIL : roy@rknec.edu

MOBILE NO. : 9692086015

NAME : Paritosh

COMPANY : Wipro

GENDER : Male

CGPA : 9.5

BRANCH : CSE

PACKAGE : 100000.0

EMAIL : dandekar@rknec.edu

MOBILE NO. : 8954205715

==============PLACEMENT STATISTICS===================

STUDENTS PLACED IN TCS = 3

STUDENTS PLACED IN INFOSYS = 6

STUDENTS PLACED IN WIPRO = 5

STUDENTS PLACED IN ACCENTURE = 2

STUDENTS PLACED IN INFORMATICA = 4

================STUDENTS STATISTICS=====================

FEMALE STUDENTS PLACED = 7

MALE STUDENTS PLACED = 12

================BRANCH STATISTICS=====================

STUDENTS PLACED FROM CSE = 10

STUDENTS PLACED FROM IT = 4

STUDENTS PLACED FROM EC = 5

**--------------------------------------------------END-----------------------------------------------------------**