1. **Write a Lex program to detect whether an input string contains alphabets or digits.**

**Source Code:-**

%{

// Macros for C program

%}

digit [0-9]

letters [a-zA-Z]

%%

{digit}+ {printf ("digit(s)");};

{letters}+ {printf ("letter(s)");};

.\* {printf ("other characters");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

231

digit(s)

fwf

letter(s)

\*%

other characters

^C

1. **Write a Lex program to check whether the input string contains only alphabets or not.**

**Source:-**

%{

// Macros for C program

%}

letter [A-Za-z]

%%

{letter}+ {printf ("Letters\n");};

.\* {printf ("Characters other than letters\n");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

abcd

Letters

aq213

Character other than letters

^C

1. **Write a Lex program to check whether the input contains only letters or only digits or both.**

Source:-

%{

// Macros for C program

%}

digit [0-9]

letters [a-zA-Z]

%%

{digit}+ {printf ("digit(s)\n");};

{letters}+ {printf ("letter(s)\n");};

{digit}+{letters}+ {printf ("letter(s) and digit(s) both\n");};

.\* {printf ("other characters");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

1234

digit(s)

abcd

letter(s)

1234abcd

digit(s) and letter(s) both

gyg\*\*(

other character

^C

1. **Write a Lex program to check whether the input string contains lower-case or upper-case letters.**

**Source:-**

%{

// Macros for C program

%}

lower [a-z]

upper [A-Z]

both [a-zA-Z]

%%

{lower}+ {printf("lowercase\n");};

{upper}+ {printf("uppercase\n");};

{both}+ {printf("Both\n");};

.\* {printf("Other characters\n");};

%%

int main()

{

yylex();

return 0;

}

int yywrap()

{

return 0;

}

**Output:-**

abcd

Lowercase

ABCD

Uppercase

AaBbCcDd

Both

1qws2

other characters

^C

1. **Write a Lex program to check whether the input is a digit or not.**

**Source:-**

%{

// Macros for C program

%}

digit [0-9]

%%

{digit} {printf ("digit");};

.\* {printf ("Not digit");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

13413

Not digit

1

Digit

^C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

1. **Write a Lex program to detect whether a string contains consonants or vowels.**

**Source:-**

%{

// Macros for C program

%}

vowel [aeiouAEIOU]

cons [b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z]

both [a-zA-Z]

%%

{vowel}+ {printf ("Contains only vowels");};

{cons}+ {printf ("Contains only consonants");};

{both}+ {printf ("Contains both vowels & consonants");};

.\* {printf ("Contains other characters");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

aeiuo

Contains only vowels

qwrtyp

Contains only consonants

qazwerty

Contains both vowels and consonants

qedc314

Contains others characters

^C

1. **Write a Lex program to count the number of characters in an input string.**

**Source:-**

%{

// Macros for C program

#include <stdio.h>

int ch = 0;

%}

%%

. {ch++;};

\n {printf("No. of Characters: %d\n\n", ch); ch = 0;};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

Hello, World!

No. of Characters: 13

^C

1. **Write a lex program to count number of vowels and consonants in the input string.**

**Source:-**

%{

// Macros for C program

#include <stdio.h>

int nv = 0, nc = 0;

%}

vowel [aeiouAEIOU]

cons [b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z]

%%

{vowel} {nv++;};

{cons} {nc++;};

\n {return 0;};

.

%%

int main() {

yylex(); //it take input from terminal

printf("No. of Vowels: %d\nNo. of Consonants: %d\n\n", nv, nc);

}

int yywrap(void)

{

return 0;

}

**Output:-**

Hello, World!

No. of vowels: 3

No. of consonants: 7

1. **Write a Lex program to count number of characters, white spaces, tabs in a string.**

**Source:-**

%{

// Macros for C program

#include <stdio.h>

int nc = 0, nt = 0, nw = 0;

%}

wspace [ ]

%%

\t {nt++;};

{wspace} {nw++;};

. {nc++;};

\n {return 0;};

%%

int main() {

yylex();

printf("No. of Characters: %d\nNo. of Whitespaces: %d\nNo. of Tabs: %d\n\n", nc, nw, nt);

}

int yywrap(void)

{

return 0;

**}**

**Output:-**

Goodmorning to one and all! Hello World!

No. of characters: 36

No. of whitespace: 6

No. of tabs: 1

1. **Write a program to check whether the input string is a keyword or not.**

**Source:-**

%{

// Program to check if input word

// is a keyword or not

%}

keywords (if|else|for|do|while|break|continue|switch|case|default|int|char)

%%

{keywords}+ {ECHO; printf(" is a keyword\n\n");};

.\* {ECHO; printf(" is not a keyword\n\n");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

If

If is a keyword

^C

1. **Write a program to check whether the input string is a valid identifier or not.**

**Source:-**

%{

//Valid Identifiers

%}

id [a-zA-Z\_][a-zA-Z\_0-9]\*

%%

{id} {ECHO; printf(" is a Valid Identifier\n\n");};

.+ {ECHO; printf(" is an invalid Identifier\n\n");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

Employee

Employee is a valid identifier

^C

1. **Write a program to check whether the given input is a valid operator or not.**

**Source:-**

%{

// Program to check valid operators

%}

ari\_operators [+\-\*\/%&\|^<>]|"<="|">="|"++"|"--"

log\_operators "&&"|"||"|"!"

%%

{ari\_operators} {ECHO; printf(" is an arithmetic operator\n");};

{log\_operators} {ECHO; printf(" is a logical operator\n");};

.\* {ECHO; printf(" is not a valid operator\n");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

+

+ is an arithmetic operator

^C

1. **Write a program to check whether the given input string is an integer, floating-point number or none of these.**

**Source:-**

%{

// Program to check if input is

// an integer, float or none

%}

digit [0-9]

integer {digit}+

floating {digit}+"."{digit}+

%%

{integer} {ECHO; printf(" is an integer\n");};

{floating} {ECHO; printf(" is a floating-point number\n");};

.\* {ECHO; printf(" is neither an integer nor a floating point number.\n");};

%%

int main() {

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

10

10 is an integer

^C

1. **Write a Lex program to count number of lexemes.**

**Source:-**

%{

// Program to find number of

// lexemes

int n\_total = 0;

%}

keywords "while"|"if"|"else"|"int"|"float"

id [a-zA-Z\_][a-zA-Z0-9\_]\*

operators "<="|"=="|"="|"++"|"-"|"\*"|"+"

seperators [(){}|, ;]

floats [0-9]\*"."[0-9]+

integers [0-9]+

%%

{keywords} {n\_total++;};

{id} {n\_total++;};

{operators} {n\_total++;};

{seperators} {n\_total++;};

{floats} {n\_total++;};

{integers} {n\_total++;};

\n {return 0;}

. ;

%%

int main() {

yylex();

printf("Total no. of lexemes = %d\n", n\_total);

}

int yywrap(void)

{

return 0;

}

**Output:-**

int a=10

Total no. of lexemes=10

1. **Write a Lex program to count total number of digit, number, words, operators, command lines, spaces, etc.**

**Source:-**

%{

// Program to find number of

// digit, number, words, operators, command lines, spaces, etc.

int nids = 0, nkws = 0, nops = 0, nfls = 0, nsep = 0, nint = 0, n\_total = 0;

%}

keywords "while"|"if"|"else"|"int"|"float"

id [a-zA-Z\_][a-zA-Z0-9\_]\*

operators "<="|"=="|"="|"++"|"-"|"\*"|"+"

seperators [(){}|, ;]

floats [0-9]\*"."[0-9]+

integers [0-9]+

%%

{keywords} {nkws++; n\_total++;};

{id} {nids++; n\_total++;};

{operators} {nops++; n\_total++;};

{seperators} {nsep++; n\_total++;};

{floats} {nfls++; n\_total++;};

{integers} {nint++; n\_total++;};

\n {return 0;}

. ;

%%

int main() {

yylex();

printf("\nNo. of identifiers = %d\n", nids);

printf("No. of keywords = %d\n", nkws);

printf("No. of operators = %d\n", nops);

printf("No. of seperators = %d\n", nsep);

printf("No. of floats = %d\n", nfls);

printf("No. of integers = %d\n", nint);

printf("Total no. of lexemes = %d\n", n\_total);

}

int yywrap(void)

{

return 0;

}

**Output:-**

int a=b\*10

No. of identifiers=2

No. of keywords=1

No. of operators=2

No. of separators=6

No. of floats=0

No. of integers=1

Total no. of lexemes=12

1. **Write a Lex code to identify if a string is a valid URL or not.**

**Source:-**

%{

// Program to check if given input

// is a valid URL or not

%}

protocol "http"|"https"|"ftp"

domain "com"|"in"|"org"|"edu"

domain\_name [a-zA-Z0-9]+

url ({protocol}:\/\/"www"\.{domain\_name}\.{domain})

%%

{url} {printf("Valid URL\n");};

.+ {printf("invalid URL\n");};

%%

int main() {

printf("Enter a url: ");

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

Enter a url: <http://www.abc.in>

Valid url

^C

1. **Write a LEX code to identify if a string is a valid Mobile Number or not.**

**Source:-**

%{

%}

ph\_no [1-9][0-9]{9}

c\_code "+91 "

%%

{c\_code}{ph\_no} {printf("Valid mobile number\n");};

.+ {printf("invalid mobile number\n");};

%%

int main() {

printf("Enter mobile number: ");

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

Enter mobile no: +916260887891

Valid mobile number

^C

1. **Write a LEX code to identify if a string is a valid email-id or not.**

**Source:-**

%{

%}

domain ".com"|".in"|".org"|".edu"

u\_name [a-z.0-9\_]+{6}

domain\_name [a-zA-Z0-9]+

email ({u\_name}"@"{domain\_name}{domain})

%%

{email} {printf("Valid email address\n");};

.+ {printf("invalid email address\n");};

%%

int main() {

printf("Enter email address: ");

yylex();

}

int yywrap(void)

{

return 0;

}

**Output:-**

Enter email address: [qwt.rye@gmail.com](mailto:qwt.rye@gmail.com)

Valid email address

^C

1. **Write a LEX code to count the number of characters in a file.**

**Source:-**

%{

// Macros for C program

#include <stdio.h>

int ch = 0;

%}

%%

. {ch++;};

%%

int yywrap(void)

{

return 1;

}

int main(int argc, char\* argv[]) {

yyin = fopen(argv[1], "r");

yylex();

printf("No. of Characters: %d\n\n", ch);

return 0;

}

**Output:**

No. of characters: 12

1. **Write a Lex program to print the total characters, white spaces, tabs and lines in the given input file.**

**Source:**

%{

// Macros for C program

#include <stdio.h>

int nc = 0, nt = 0, nw = 0, nl = 0;

%}

wspace [ ]

%%

\n {nl++;};

\t {nt++;};

{wspace} {nw++;};

. {nc++;};

%%

int main(int argc, char\* argv[]) {

yyin = fopen(argv[1], "r");

yylex();

fclose(yyin);

printf("No. of Characters: %d\nNo. of Whitespaces: %d\nNo. of Tabs: %d\nNo. of lines: %d\n", nc, nw, nt, nl);

}

int yywrap(void)

{

return 1;

}

**Output:**

Hello

My name is Praveen

I am student of IIIT Ranchi

No. of characters:43

No. of whitespaces :9

No. of tabs:0

No. of lines:3

1. **Write a Lex program to count number of lexemes in the given input file.**

**Source:-**

%{

// Program to find number of

// lexemes

int n\_total = 0;

%}

keywords "while"|"if"|"else"|"int"|"float"

id [a-zA-Z\_][a-zA-Z0-9\_]\*

operators "<="|"=="|"="|"++"|"-"|"\*"|"+"

seperators [(){}|, ;]

floats [0-9]\*"."[0-9]+

integers [0-9]+

%%

{keywords} {n\_total++;};

{id} {n\_total++;};

{operators} {n\_total++;};

{seperators} {n\_total++;};

{floats} {n\_total++;};

{integers} {n\_total++;};

\n {return 0;}

. ;

%%

int main(int argc, char\* argv[]) {

yyin = fopen(argv[1], "r");

yylex();

printf("Total no. of lexemes = %d\n", n\_total);

}

int yywrap(void)

{

return 1;

}

**Output:**

int a=10

no. of lexemes: 8

1. **Write a Lex program to take input from file and remove multiple spaces, newline and tab and write output in a separate file.**

**Source:**

%{

// Macros for C program

#include <stdio.h>

int s = 0, nl = 0;

%}

wspace [ ]

%%

\n {if (!nl) {

fprintf(yyout, "\n");

nl = 1;

}

else

fprintf(yyout, "");};

\t {fprintf(yyout, "");};

{wspace} {

if (!s) {

fprintf(yyout, " ");

s = 1;

}

else

fprintf(yyout, "");

};

. {fprintf(yyout, "%s", yytext);s=0;nl=0;};

%%

int main(int argc, char\* argv[]) {

yyin = fopen(argv[1], "r");

yyout = fopen("op", "w");

yylex();

fclose(yyin);

}

int yywrap(void)

{

return 1;

}

1. **Trying out Bison to write a Hello World program**

**Source:**

%{

#include "y.tab.h"

int yyerror(char \*errormsg);

int yyparse();

%}

%%

("hi"|"oi"|"hello")"\n" { return HI; }

("ttyl"|"bye")"\n" { return BYE; }

[-[]+.,><] { return yytext[0]; }

. { yyerror("Unknow char"); }

%%

int main(void)

{

yyparse();

return 0;

}

int yywrap(void)

{

return 0;

}

int yyerror(char \*errormsg)

{

fprintf(stderr, "%s\n", errormsg);

exit(1);

}

**Output:**

Hello

Hello world

Ttyl

Bye world

1. **Write a program using Lex and YACC to check whether given string is Palindrome or not.**

**Source:**

/\* lex program for recognition of palindrome \*/

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

/\* Generated by yacc \*/

#include "y.tab.h"

%}

letters [a-zA-Z]

%%

{letters}+ {yylval.string = yytext; return STR;}

\n {return newline;}

[ \t] ;

. {ECHO; yyerror("Unexpected Character");}

%%

int yywrap() {

return 1;

}

**Output:-**

Malyalam

Palindrome

^C

1. **Write a program using Lex and YACC which accept strings that starts and ends with 0 or 1**

**Source:-**

%{

//lex file to check whether the input

//string ends with 0/1 or not

# include <stdio.h>

# include <stdlib.h>

# include "y.tab.h"

%}

%%

[a-zA-Z0-9" "\t]+ {

yylval.f = yytext;

return STR;

}

[\n] {;}

.\* {return yytext[0];}

%%

int yywrap(){

return -1;

}

**Output:**

11011

The input string is accepted