**Program to Check for Capital Letters**

%{

int count = 0;

%}

%%

[A-Z] { printf("%s capital letter\n", yytext);

count++; }

. { printf("%s not a capital letter\n", yytext); }

\n { return 0; }

%%

int yywrap() { return 1; }

int main() {

yylex();

printf("\nNumber of Capital letters in the given input - %d\n", count);

return 0;

}

**2) Program to Count the Number of Lines and Characters**

%{

int no\_of\_lines = 0;

int no\_of\_chars = 0;

%}

%%

\n { ++no\_of\_lines; }

. { ++no\_of\_chars; }

%%

int yywrap() { return 1; }

int main() {

yylex();

printf("Number of lines = %d, Number of characters = %d\n", no\_of\_lines, no\_of\_chars);

return 0;

}

**Lex Program to Count the Total Number of Characters**

%{

int char\_count = 0;

%}

%%

. { char\_count++; } // Increment count for every character

\n { char\_count++; } // Count newline characters too

%%

int yywrap() { return 1; }

int main() {

yylex();

printf("Total number of characters: %d\n", char\_count);

return 0;

}

**Lex Program to Count the Number of Words**

%{

int word\_count = 0;

%}

%%

[^\t\n ]+ { word\_count++; } // Match any non-space character sequence as a word

\n { } // Ignore newlines

%%

int yywrap() { return 1; }

int main() {

yylex();

printf("Total number of words: %d\n", word\_count);

return 0;

}

**A simple lexer that recognizes iden􀆟fiers, numbers, and operators.**

Code (lexer.l):

%{

#include <stdio.h>

%}

%%

[a-zA-Z\_][a-zA-Z0-9\_]\* { prin􀆞("IDENTIFIER: %s\n", yytext); }

[0-9]+ { prin􀆞("NUMBER: %s\n", yytext); }

[+\-\*/=] { prin􀆞("OPERATOR: %s\n", yytext); }

. { prin􀆞("OTHER: %s\n", yytext); }

%%

int main() {

yylex();

return 0;

}

int yywrap() { return 1; }

**Lex Program to Count the Number of Lines, Spaces, and Tabs**

%{

int line\_count = 0, space\_count = 0, tab\_count = 0;

%}

%%

\n { line\_count++; } // Count new lines

" " { space\_count++; } // Count spaces

"\t" { tab\_count++; } // Count tabs

%%

int yywrap() { return 1; }

int main() {

yylex();

printf("Total lines: %d\n", line\_count);

printf("Total spaces: %d\n", space\_count);

printf("Total tabs: %d\n", tab\_count);

return 0;

}

**Simple calculator program**

%{

#include <stdio.h>

#include <stdlib.h>

int yylval;

%}

%%

[0-9]+ { yylval = atoi(yytext); return 'n'; } // Recognize numbers

[+\-\*/()] { return yytext[0]; } // Operators and parentheses

\n { return 0; } // End of input

[ \t] { } // Ignore spaces and tabs

. { printf("Invalid character: %s\n", yytext); } // Handle invalid characters

%%

int yywrap() { return 1; }

**Calculator Program for Lex and Bison**

Step 1: Create the Lexer (calc.l)

%{

#include "y.tab.h"

%}

%%

[0-9]+ { yylval = atoi(yytext); return NUMBER; }

[+\-\*/] { return yytext[0]; }

\n { return 0; }

. { return yytext[0]; }

%%

int yywrap() { return 1; }

Step 2: Create the Parser (calc.y)

%{

#include <stdio.h>

#include <stdlib.h>

%}

%token NUMBER

%%

expression: expression '+' expression { prin􀆞("%d\n", $1 + $3); }

| expression '-' expression { prin􀆞("%d\n", $1 - $3); }

| expression '\*' expression { prin􀆞("%d\n", $1 \* $3); }

| expression '/' expression { prin􀆞("%d\n", $1 / $3); }

| NUMBER { $$ = $1;

} ;

%%

int main() {

yyparse();

return 0;

}

void yyerror(const char \*s) {

prin􀆞("Error: %s\n", s);

}

How to Compile and Run:

bison -d calc.y

lex calc.l

gcc lex.yy.c calc.tab.c -o calc -ll

./calc

Enter expressions like 5+3, and it will evaluate them.