**1. Problem name: Write a Simple lex specification to recognize different key words**

**Source Code:**

%{

#include <stdio.h>

%}

%%

if { printf("Keyword: if\n"); } else { printf("Keyword: else\n"); }

while { printf("Keyword: while\n"); } for { printf("Keyword: for\n"); } do { printf("Keyword: do\n"); } switch { printf("Keyword: switch\n"); }

case { printf("Keyword: case\n"); }

default { printf("Keyword: default\n"); }

return { printf("Keyword: return\n"); }

break { printf("Keyword: break\n"); }

int { printf("Keyword: int\n"); }

float { printf("Keyword: float\n"); }

double { printf("Keyword: double\n"); }

char { printf("Keyword: char\n"); }

void { printf("Keyword: void\n"); }

bool { printf("Keyword: bool\n"); }

const { printf("Keyword: const\n"); }

static { printf("Keyword: static\n"); }

struct { printf("Keyword: struct\n"); }

typedef { printf("Keyword: typedef\n"); }

%%

int main() {

printf("Enter some code to recognize keywords:\n");

yylex();

return 0;

}

**2. Problem name: Write a Simple lex specification to recognize the identifier**

**Scource code:**

%{

#include <stdio.h>

%}

%%

/\* Identifiers: Start with a letter or underscore, followed by letters, digits, or underscores \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

%%

int main() {

printf("Enter some code to recognize identifiers:\n");

yylex();

return 0;

}

**3. Problem name: Write a Simple lex specification to recognize Real Number**

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Real numbers: floating-point numbers with optional fractional part \*/

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

/\* Integer numbers (optional positive or negative sign) \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter numbers :\n");

yylex();

return 0;

}

**4. Problem name: Write a Simple lex specification to recognize Integer**

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Integer: A sequence of digits with an optional + or - sign \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter numbers:\n");

yylex();

return 0;

}

**5. Problem name:** **Write a Simple lex specification to recognize Float**

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Floating-point number: an optional sign followed by digits, a decimal point, and more digits \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

/\* Ignore any other characters \*/

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter numbers to recognize floats:\n");

yylex();

return 0;

}

**6. Problem Name:** Write a Simple lex specification to recognize for the positive and negative integer and float number

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Positive Float: a sequence of digits followed by a decimal point and more digits \*/

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

/\* Negative Float: a negative sign followed by a sequence of digits, a decimal point, and more digits \*/

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

/\* Positive Integer: a sequence of digits \*/

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

/\* Negative Integer: a negative sign followed by a sequence of digits \*/

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

/\* Positive Integer without a sign \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

/\* Ignore any other characters \*/

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter numbers to recognize positive and negative integers and floats:\n");

yylex();

return 0;

}

**7. Problem Name: Write a Simple lex specification to recognize Different Punctuation Symbol**

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Match different punctuation symbols \*/

[.,!?;:()-] { printf("Punctuation: %s\n", yytext); }

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

/\* Ignore any other characters \*/

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter text to recognize punctuation symbols:\n");

yylex();

return 0;

}

**8. Problem Name: Write a Simple lex specification to recognize digit**

**Source Code:**

%{

#include <stdio.h>

%}

%%

/\* Match a single digit (0-9) \*/

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

/\* Ignore whitespace \*/

[ \t\n] { /\* Ignore whitespace \*/ }

/\* Ignore any other characters \*/

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter text to recognize digits:\n");

yylex();

return 0;

}

**9. recognize Operator**

%{

#include <stdio.h>

%}

%%

/\* Arithmetic Operators \*/

"++"|"--"|"+"|"-"|"\*"|"/"|"%" { printf("Arithmetic Operator: %s\n", yytext); }

"=="|"!="|"<"|">"|"<="|">=" { printf("Relational Operator: %s\n", yytext); }

"&&"|"||"|"!" { printf("Logical Operator: %s\n", yytext); }

"&"|"|"|"^"|"<<"|">>" { printf("Bitwise Operator: %s\n", yytext); }

"="|"+="|"-="|"\*="|"/="|"%=" { printf("Assignment Operator: %s\n", yytext); }

[ \t\n] { /\* Ignore whitespace \*/ }

. { /\* Ignore any other characters \*/ }

%%

int main() {

printf("Enter a mathematical expression or sentence to recognize operators:\n");

yylex();

return 0;

}

int yywrap() {

return 1;

}

**10. Problem Name: write a simple lex specification to recognize the following verbs: is, am, are, were, was, be, being, been, do ,does, did will would should can, could, has, have, had, go.**

**Source Code:**

%{

#include <stdio.h>

%}

%%

is|am|are|were|was|be|being|been|do|does|did|will|would|should|can|could|has|have|had|go {

printf("VERB FOUND: %s\n", yytext);

}

\n ; /\* Ignore newlines \*/

[ \t]+ ; /\* Ignore whitespaces \*/

. ; /\* Ignore other characters \*/

%%

int main() {

printf("Enter a sentence: ");

yylex(); // Start scanning

return 0;

}

int yywrap() {

return 1;

}

**Result:**