4)**NO.OF VOWEL ND CONS**

%{

#include <stdio.h>

/\* Declare global counters \*/

int vowel = 0, cons = 0;

%}

%%

[aeiouAEIOU] { ++vowel; } /\* Increment vowel count for vowels \*/

[a-zA-Z] { ++cons; } /\* Increment consonant count for consonants \*/

.|\n { /\* Ignore other characters \*/ }

%%

int yywrap() {

return 1;

}

int main() {

yylex(); /\* Start lexical analysis \*/

printf("Vowel count: %d\n", vowel);

printf("Consonant count: %d\n", cons);

return 0;

}

5)**KEYWORD AND IDENTIFIER**

%{

int key=0,iden=0;

%}

%%

if|else|while|do|for {printf("keywords:%s\n",yytext);++key;}

[a-zA-Z\_0-9]+ {printf("identifier:%s\n",yytext);}

.|\n ;

%%

int yywrap(){

return 1;

}

int main(){

yylex();

printf("key count is:%d\n",key);

return 0;

}

6) **POSITIVE AND NEGATIVE**

%{

int po=0,ne=0;

%}

%%

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

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

%%

int yywrap(){

return 1;

}

int main(){

yylex();

printf("=ve count is:%d\n",po);

return 0;

}

8)**START WITH VOWEL**

%%

[aeiouAEIOU][a-zA-Z]+ {printf("accepted");}

[a-zA-Z]+ {printf("not accepted");}

.|\n ;

%%

int yywrap(){

return 1;

}

int main(){

yylex();

return 0;

}

10) %%

https://www\.[a-zA-Z0-9]+\.[a-zA-Z]{2,} {printf("valid\n");}

.|\n {printf("not valid\n");}

%%

int yywrap() {

return 1;

}

int main() {

yylex();

return 0;

}

15) **VALID EMAIL**

%%

[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,} {printf("Valid email\n");}

.|\n {printf("Not valid email\n");}

%%

int yywrap() {

return 1;

}

int main() {

yylex();

return 0;

}

11)**VALIDD DOB**

%%

[0-3][0-9]/[0-1][0-9]/[0-9]{4} { printf("Valid DOB: %s\n", yytext); }

.|\n { /\* Ignore other input \*/ }

%%

int main() {

yylex();

return 0;

}

14) **BASIC OP**

%{

#include<stdio.h>

float op1=6,op2=7;

%}

%%

"+" {printf("sum =%f",op1+op2);}

"-" {printf("diff=%f",op1-op2);}

"\*" {printf("mul=%f",op1\*op2);}

"/" {printf("div=%f",op1/op2);}

%%

int yywrap(){}

int main()

{

printf("enter proper operator.");

yylex();

}

17)**NO OF LINES**

%{

#include<stdio.h>

int l=0,w=0,c=0;

%}

%%

[\n] {++l;}

[ ] {++w;}

[a-zA-Z0-9] {++c;}

%%

int yywrap(){

return 1;

}

int main(){

yylex();

printf("word:%d",w);

printf("char:%d",c);

printf("limes:%d",l);

return 0;

}

18)**PRINT CONSTANTS**

%%

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

[0-9]+ {printf("int constant:%s",yytext);}

.|\n ;

%%

int yywrap(){

return 1;

}

int main(){

yylex();

return 0;

}

19)**MACROS**

%{

#include<stdio.h>

int m=0,h=0;

%}

%%

"#define" {++m;}

"#include" {++h;}

%%

int yywrap(){

return 1;

}

int main(){

yylex();

printf("header count is:%d\n",h);

return 0;

}

21)**ADD LINES**

%{

#include<stdio.h>

int line = 1;

%}

%%

.\* { printf("%d: %s", line++, yytext); }

%%

int yywrap() { return 1; }

int main() { yylex(); return 0; }

22) **COMMENT LINES**

%{

#include<stdio.h>

int c = 0;

%}

%%

"//".\* { ++c; } /\* Match single-line comments \*/

"/\*".\*"\*/" { ++c; } /\* Match multi-line comments \*/

.|\n ; /\* Ignore all other input \*/

%%

int yywrap() {

return 1;

}

int main() {

yylex();

printf("Comment lines are: %d\n", c);

return 0;

}

24)**TOKENS**

%{

#include<stdio.h>

%}

%%

"int"|"float"|"char"|"double"|"void" { printf("Keyword: %s\n", yytext); }

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

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

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

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

"(" | ")" | "{" | "}" | ";" { printf("Symbol: %s\n", yytext); }

.|\n ; /\* Ignore other inputs \*/

%%

int yywrap() {

return 1;

}

int main() {

printf("Enter the program code:\n");

yylex();

return 0;

}

13) %{

#include <stdio.h>

%}

%%

"apple" { printf("orange"); }

.|\n { printf("%s", yytext); }

%%

int yywrap() {

return 1;

}

int main() {

yylex();

return 0;

}

9)**MAXLENGTH \**

%{

#include <stdio.h>

int maxLength = 0;

%}

20)

%%

[a-zA-Z]+ { if (yyleng > maxLength) maxLength = yyleng; }

.|\n { /\* Ignore non-words \*/ }

%%

int yywrap() {

return 1;

}

int main() {

yylex();

printf("Length of the longest word: %d\n", maxLength);

return 0;

}

**HTML TAG**

%{

int tags;

%}

%%

"<"[^>]\*> { tags++; printf("%s \n", yytext); }

.|\n { }

%%

int yywrap(void) {

return 1; }

int main(void)

{

FILE \*f;

char file[10];

printf("Enter File Name : ");

scanf("%s",file);

f = fopen(file,"r");

yyin = f;

yylex();

printf("\n Number of html tags: %d",tags);

fclose(yyin);

}

**C progs:**

**1)separate tokens**

**#include <stdio.h>**

**int main() {**

**char input[100];**

**char \*token;**

**// Get input from the user**

**printf("Enter an expression: ");**

**fgets(input, sizeof(input), stdin);**

**// Loop through each character of the input**

**printf("Tokens: \n");**

**for (int i = 0; input[i] != '\0'; i++) {**

**// Check for letters (Identifiers)**

**if ((input[i] >= 'a' && input[i] <= 'z') || (input[i] >= 'A' && input[i] <= 'Z')) {**

**printf("Identifier: %c\n", input[i]);**

**}**

**// Check for digits (Constants)**

**else if (input[i] >= '0' && input[i] <= '9') {**

**printf("Constant: %c\n", input[i]);**

**}**

**// Check for operators**

**else if (input[i] == '+' || input[i] == '-' || input[i] == '\*' || input[i] == '/' || input[i] == '=' || input[i] == '<' || input[i] == '>') {**

**printf("Operator: %c\n", input[i]);**

**}**

**}**

**return 0;**

**}**

**2)comment or not**

**#include <stdio.h>**

**#include <string.h>**

**int main() {**

**char line[100];**

**// Get input from the user**

**printf("Enter a line of code: ");**

**fgets(line, sizeof(line), stdin);**

**// Check for single-line comment**

**if (line[0] == '/' && line[1] == '/') {**

**printf("This is a single-line comment!\n");**

**}**

**// Check for multi-line comment**

**else if (line[0] == '/' && line[1] == '\*' && line[strlen(line) - 2] == '\*' && line[strlen(line) - 1] == '/') {**

**printf("This is a multi-line comment!\n");**

**}**

**else {**

**printf("This is not a comment.\n");**

**}**

**return 0;**

**}**

**3)operator identify**

**#include <stdio.h>**

**int main() {**

**char ch;**

**// Get input from the user**

**printf("Enter an operator: ");**

**ch = getchar();**

**// Check for each operator using if-else conditions**

**if (ch == '+') {**

**printf("This is an Addition operator!\n");**

**} else if (ch == '-') {**

**printf("This is a Subtraction operator!\n");**

**} else if (ch == '\*') {**

**printf("This is a Multiplication operator!\n");**

**} else if (ch == '/') {**

**printf("This is a Division operator!\n");**

**} else if (ch == '=') {**

**printf("This is an Assignment operator!\n");**

**} else if (ch == '<') {**

**printf("This is a Less-than operator!\n");**

**} else if (ch == '>') {**

**printf("This is a Greater-than operator!\n");**

**} else {**

**printf("This is not an Operator.\n");**

**}**

**return 0;**

**}**

**5)no.of words ,lines,chars**

**#include <stdio.h>**

**int main() {**

**char ch;**

**int words = 0, chars = 0, lines = 0;**

**int inWord = 0; // Flag to track if we're inside a word**

**// Get input until EOF (Ctrl+D) or newline**

**printf("Enter a sentence (Ctrl+D to end input):\n");**

**while ((ch = getchar()) != EOF) {**

**chars++; // Count every character**

**if (ch == '\n') {**

**lines++; // Count new lines**

**}**

**// Count words: if space or punctuation is encountered, a word ends**

**if (ch == ' ' || ch == '\n' || ch == '\t') {**

**inWord = 0;**

**} else if (inWord == 0) {**

**inWord = 1; // A new word has started**

**words++;**

**}**

**}**

**// Output the counts**

**printf("\nWords: %d\n", words);**

**printf("Characters: %d\n", chars);**

**printf("Lines: %d\n", lines);**

**return 0;**

**}**

**6)identifier valid or not**

**#include <stdio.h>**

**int main() {**

**char identifier[100];**

**int valid = 1; // Assume valid until proven otherwise**

**// Get input from the user**

**printf("Enter an identifier: ");**

**scanf("%s", identifier);**

**// Check if the first character is a letter or underscore**

**if ((identifier[0] >= 'a' && identifier[0] <= 'z') || (identifier[0] >= 'A' && identifier[0] <= 'Z') || identifier[0] == '\_') {**

**// Check the rest of the characters**

**for (int i = 1; identifier[i] != '\0'; i++) {**

**if (!( (identifier[i] >= 'a' && identifier[i] <= 'z') ||**

**(identifier[i] >= 'A' && identifier[i] <= 'Z') ||**

**(identifier[i] >= '0' && identifier[i] <= '9') ||**

**identifier[i] == '\_')) {**

**valid = 0; // Invalid if any character doesn't match**

**break;**

**}**

**}**

**} else {**

**valid = 0; // Invalid if the first character is not a letter or underscore**

**}**

**// Output result**

**if (valid) {**

**printf("Valid identifier!\n");**

**} else {**

**printf("Invalid identifier!\n");**

**}**

**return 0;**

**}**

**7)thee address code**

**#include <stdio.h>**

**int main() {**

**char var1, var2, var3, var4; // Variables for the expression**

**char operator1, operator2; // Operators for the expression**

**char temp1 = 't'; // Temporary variable to store results**

**// Get input from user for the variables and operators**

**printf("Enter variable names (e.g., a b c d): ");**

**scanf(" %c %c %c %c", &var1, &var2, &var3, &var4);**

**printf("Enter operators (e.g., + \*): ");**

**scanf(" %c %c", &operator1, &operator2);**

**// Displaying the Three-Address Code**

**printf("\nThree-Address Code (TAC):\n");**

**// Generating the code for the expression a = b + c \* d**

**printf("t1 = %c %c %c\n", var3, operator2, var4); // t1 = c \* d**

**printf("%c = %c %c t1\n", var1, var2, operator1); // a = b + t1**

**return 0;**

**}**