**COMPILER DESIGN (01CE0601)**

**2022-2023**

**STUDENT LAB MANUAL**

**INDEX**

|  |  |
| --- | --- |
| **Ex No** | **Name of Experiments** |
| 1 | WAP to verify that the given input is valid keyword. |
| 2 | WAP to verify that the given input is valid identifier. |
| 3 | Case study of Lex tool. |
| 4 | Write a Lex Program to   1. to count vowels and consonants 2. count words, characters, lines and space from the given inputs. |
| 5 | Write a Lex Program to count words, characters, lines, spaces, vowels and consonants from the given file |
| 6 | Implement Following Programs using lex.   1. Write a Lex Program to check whether the generated string is ending with zeros. 2. Write a Lex Program to check given string is simple or compound string. |
| 7 | Implement Following Programs using lex.   1. Write a Lex Program to check given number is positive negative or zero. 2. Write a Lex Program to print HTML tags of given file. |
| 8 | WA Lex Program to count the total number of printf and scanf statement in given C file. Also convert it into readf and write out respectively to another  file. |
| 9 | Implementation of Finite Automata and string validation. |

# Experiment 1

## WAP to verify that the given input is valid keyword.

#include <stdio.h> #include <string.h> int main()

{

char arr[32][10] =

{"auto","break","case","char","const","continue","default","

do","double","else","enum","extern","float","for","goto","if

","int","long","register","return","short","signed","sizeof"," static","struct","switch","typedef","union","unsigned","void ","volatile","while"};

char str[10];

printf("Enter the string to check: "); scanf("%[^\n]%\*c", str);

int i,flag=0; for(i=0;i<=32;i++){

if(strcmp(str,arr[i])==0){ flag=1;

}

}

if (flag == 1){

printf("%s is a keyword",str);

}

else{

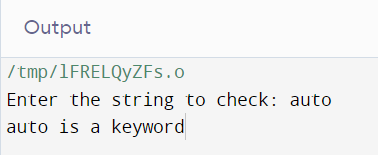
printf("%s is not a keyword",str);

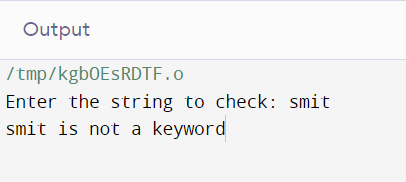
}

return 0;

}

**Output:**





# Experiment 2

## WAP to verify that the given input is valid identifier.

#include <bits/stdc++.h> using namespace std;

bool isValid(string str, int n)

{

if (!((str[0] >= 'a' && str[0] <= 'z')

|| (str[0] >= 'A' && str[0] <= 'Z')

|| str[0] == '\_')) return false;

for (int i = 1; i < str.length(); i++) { if (!((str[i] >= 'a' && str[i] <= 'z')

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

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

|| str[i] == '\_')) return false;

}

return true;

}

int main()

{

string str; cout<<"Input string: ";

cin>>str;

int n = str.length();

if (isValid(str, n)) cout << "Valid";

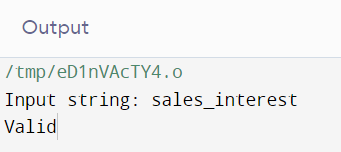
else

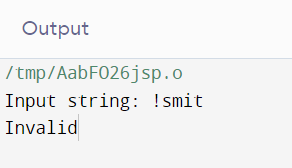
cout << "Invalid";

return 0;

}

**Output:**





# Experiment 3

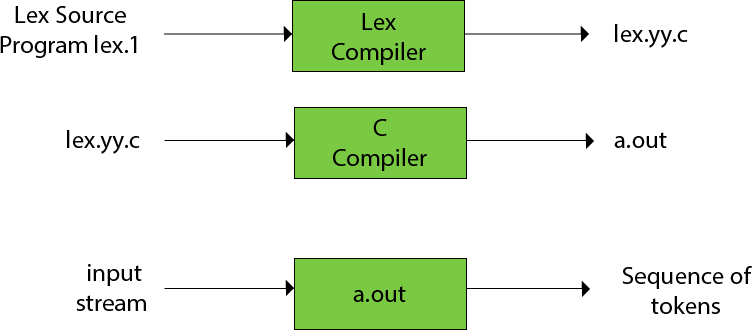
## Case study of Lex tool.

1. **Introduction: -**

* Lex is a program that generates lexical analyzer. It is used with YACC parser generator.
* The lexical analyzer is a program that transforms an input stream into a sequence of tokens.
* It reads the input stream and produces the source code as output through implementing the lexical analyzer in the C program.

1. **Steps: -**

* Firstly, lexical analyzer creates a program lex.1 in the Lex language. Then Lex compiler runs the lex.1 program and produces a C program lex.yy.c.
* Finally, C compiler runs the lex.yy.c program and produces an object program a.out.
* a.out is lexical analyzer that transforms an input stream into a sequence of tokens.



1. **Structure of program: -**

{Delceration }

%%

{Rules}

%%

{User subroutines}

1. **Installation**

**Step 1:** Download the Flex tool from the url: <http://gnuwin32.sourceforge.net/packages/flex.htm>

**Step 3:** Now check for the executable file in downloads in your system and run it. **Step 4:** It will prompt confirmation to make changes to your system. Click on Yes. **Step 5:** Setup screen will appear, click on Next.

**Step 6:** The next screen will be of License Agreement; click on I Agree.

**Step 7:** The next screen will be of installing location so choose the drive which will have sufficient memory space for installation. It needed only a memory space of

176.7 MB.

**Step 8:** Next screen will be of choosing the Start menu folder so don’t do anything just click on the Next Button.

**Step 9:** After this installation process will start and will hardly take a minute to complete the installation.

**Step 10:** Click on Finish after the installation process is complete. Keep the tick mark on the checkbox if you want to run Flex now if not then uncheck it.

**Step 11:** Flex Windows is successfully installed on the system.

# Experiment 4

## Write a Lex Program to

1. **to count vowels and consonants**

%{

int vow\_count=0; int const\_count =0;

%}

%%

[aeiouAEIOU] {vow\_count++;} [a-zA-Z] {const\_count++;}

%%

int yywrap(){} int main()

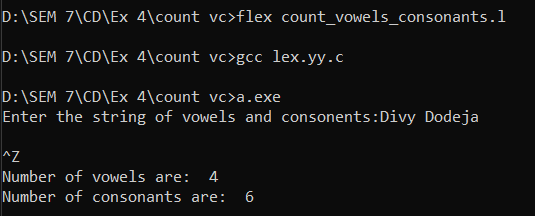
{

printf("Enter the string of vowels and consonents:"); yylex();

printf("Number of vowels are: %d\n", vow\_count); printf("Number of consonants are: %d\n", const\_count); return 0;

}

**Output**



## count words, characters, lines and space from the given inputs.

%{

#include<stdio.h>

int sc=0,wc=0,lc=0,cc=0;

%}

%%

[\n] { lc++; cc+=yyleng;} [ \t] { sc++; cc+=yyleng;}

[^\t\n ]+ { wc++; cc+=yyleng;}

%%

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

{

printf("Enter the input:\n"); yylex();

printf("The number of lines=%d\n",lc); printf("The number of spaces=%d\n",sc); printf("The number of words=%d\n",wc); printf("The number of characters are=%d\n",cc);

}

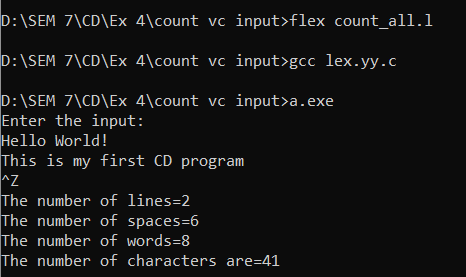
int yywrap( )

{

return 1;

}

**Output:**



# Experiment 5

## Write a Lex Program to count words, characters, lines, spaces, vowels and consonantsfrom the given file.

%{

#include<stdio.h>

int sc=0,wc=0,lc=1,cc=0, vow\_count=0,const\_count=0;

%}

%%

[aeiouAEIOU] {vow\_count++;} [a-zA-Z] {const\_count++;}

[\n] { lc++; cc+=yyleng;} [ \t] { sc++; cc+=yyleng;}

[^\t\n ] { wc++; cc+=yyleng;}

%%

int yywrap( ){return 1;} int main()

{

yyin = fopen("abc.txt","r"); yylex();

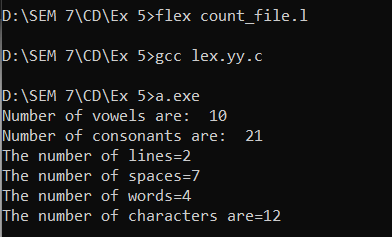
printf("Number of vowels are: %d\n", vow\_count); printf("Number of consonants are: %d\n", const\_count);

printf("The number of lines=%d\n",lc); printf("The number of spaces=%d\n",sc); printf("The number of words=%d\n",wc); printf("The number of characters are=%d\n",cc);

return 0;

}

**Output:**



# Experiment 6

## Implement Following Programs using lex.

1. **Write a Lex Program to check whether the generated string is ending with zeros.**

%{

#include<stdio.h>

%}

%%

[0+1]\*0 {printf("THE STRING GENERATED ENDS WITH 0");}

.\* {printf("STRING DOESN'T END WITH 0");}

%%

void main()

{

printf("ENTER A STRING: ");

yylex();

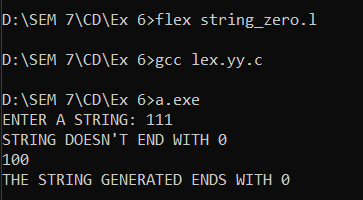
}

int yywrap()

{ return 1;

}

**Output:**



## Write a Lex Program to check given string is simple or compound string.

%{

#include<stdio.h> int simple=1;

%}

%%

[ \t]+[aA][nN][dD][ \t]+ {simple=0;}

[ \t]+[bB][uU][tT][ \t]+ {simple=0;}

[ \t]+[oO][rR][ \t]+ {simple=0;}

%%

int main()

{ printf("ENTER THE SENTENCE TO CHECK WHETHER SIMPLE OR COMPOUND: ");

yylex(); if(simple==1)

printf("\nTHE SENTENCE IS A SIMPLE SENTENCE.\n");

else

printf("\nTHE SENTENCE IS A COMPOUND SENTENCE.\n");

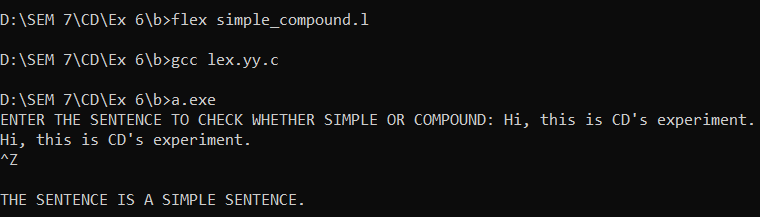
}

int yywrap()

{ return 1;

}

**Output:**



# Experiment 7

## Implement Following Programs using lex.

1. **Write a Lex Program to check given number is positive negative or zero.**

%{

#include<stdio.h> int a=0,b=0,c=0,d=0;

%}

%%

0\* printf("Zero");

[+]?[0-9]+ {printf("Postive Integer");a++;}

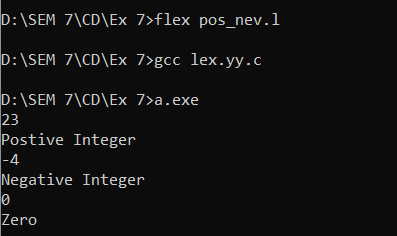
[-]{1}[0-9]+ {printf("Negative Integer");b++;}

%%

int yywrap(){} int main(){ yylex();

return 0;}

**Output:**



## Write a Lex Program to print HTML tags of given file.

%{

%}

%%

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

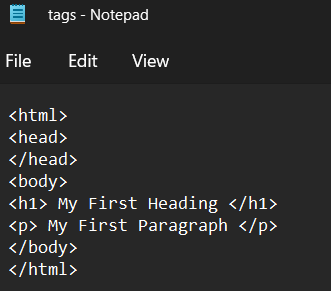
. ;

%%

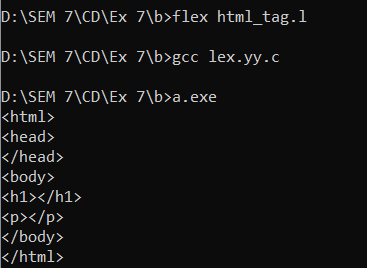
int yywrap(){} int main(){

yyin = fopen("tags.txt","r"); yylex();

return 0; }



**Output:**



# Experiment 8

## WAP Lex Program to count the total number of printf and scanf statement in given Cfile. Also convert it into readf and writeout respectively to another file.

%{

#include <stdio.h> int pf=0, sf=0;

%}

%%

"scanf" {sf++; fprintf(yyout, "readf");} "printf" {pf++; fprintf(yyout, "writeout");}

%%

void main()

{

yyin = fopen("open.c", "r");

yyout=fopen("output.c","w"); yylex();

printf("No. of printf = %d" , pf); printf("No. of scanf = %d",sf);

}

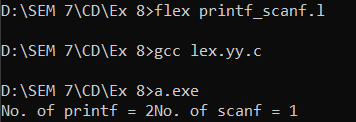
int yywrap()

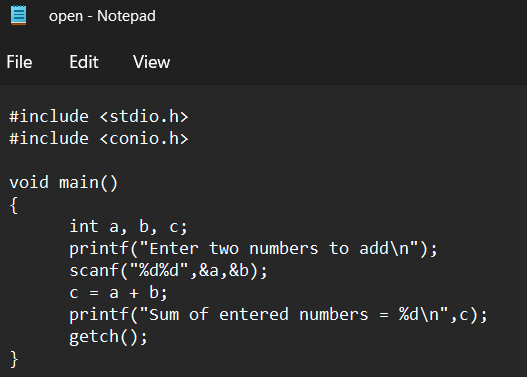
{

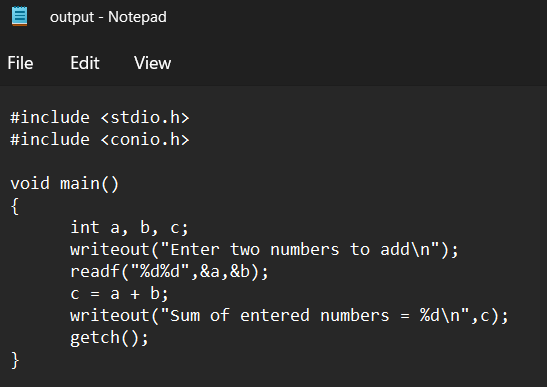
return 1;

}

**Output:**







# Experiment 9

## Implementation of Finite Automata and string validation.

#include <stdio.h> #include <string.h>

//#include <conio.h> void main()

{

char A[100];

int current\_state; int num\_states= 4;

printf("Enter a string : " ); scanf("%s", A);

if( strlen(A)== 3)

{

current\_state=1; if(current\_state==1 && A[0] == 'a')

{

current\_state=2;

}

if(current\_state==2 && A[1] == 'b')

{

current\_state=3;

}

if(current\_state==3 && A[2] == 'b')

{

current\_state=4;

}

if(current\_state==4)

{

printf("String is accepted");

}

else

{

printf("String is Rejected, Try Again \n");

}

}

else

{

printf("String is rejected. Try again");

}

}

**Output**

