****

**Department of**

**Computer Science and Engineering**

**Lab Assignment – 02**

Course No. : CSE-354

Course Title : Compiler Design Laboratory

Name of Experiment: Report on Solving Problems using Flex

Submitted By : Kamelia Zaman Moon

ID : 299

Session : 2017-18

Date : 18-03-2021

Submitted To : Tahsina Hashem

Assistant Professor

Department of Computer Science and Engineering

Jahangirnagar University, Savar.

**Problem-1:**

In this assignment, your job is to program a simple lexical analyzer which recognizes the following keywords (int, float, if, then, else, for, printf, switch, case, return) and integers, floating point numbers.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

int countInt=0;

int countFloat=0;

%}

keyword "int"|"float"|"if"|"then"|"else"|"for"|"printf"|"switch"|"case"|"return"

digit [0-9]

%%

{keyword} {cout<<"Found keyword"<<endl;}

{digit}+ {cout<<"Found integer: "<<yytext<<endl; countInt++;}

{digit}+"."{digit}+ {cout<<"Found floating point number: "<<yytext<<endl; countFloat++;}

%%

int main()

{

yylex();

cout<<"Total valid integers: "<<countInt<<endl;

cout<<"Total valid floating point numbers: "<<countFloat<<endl;

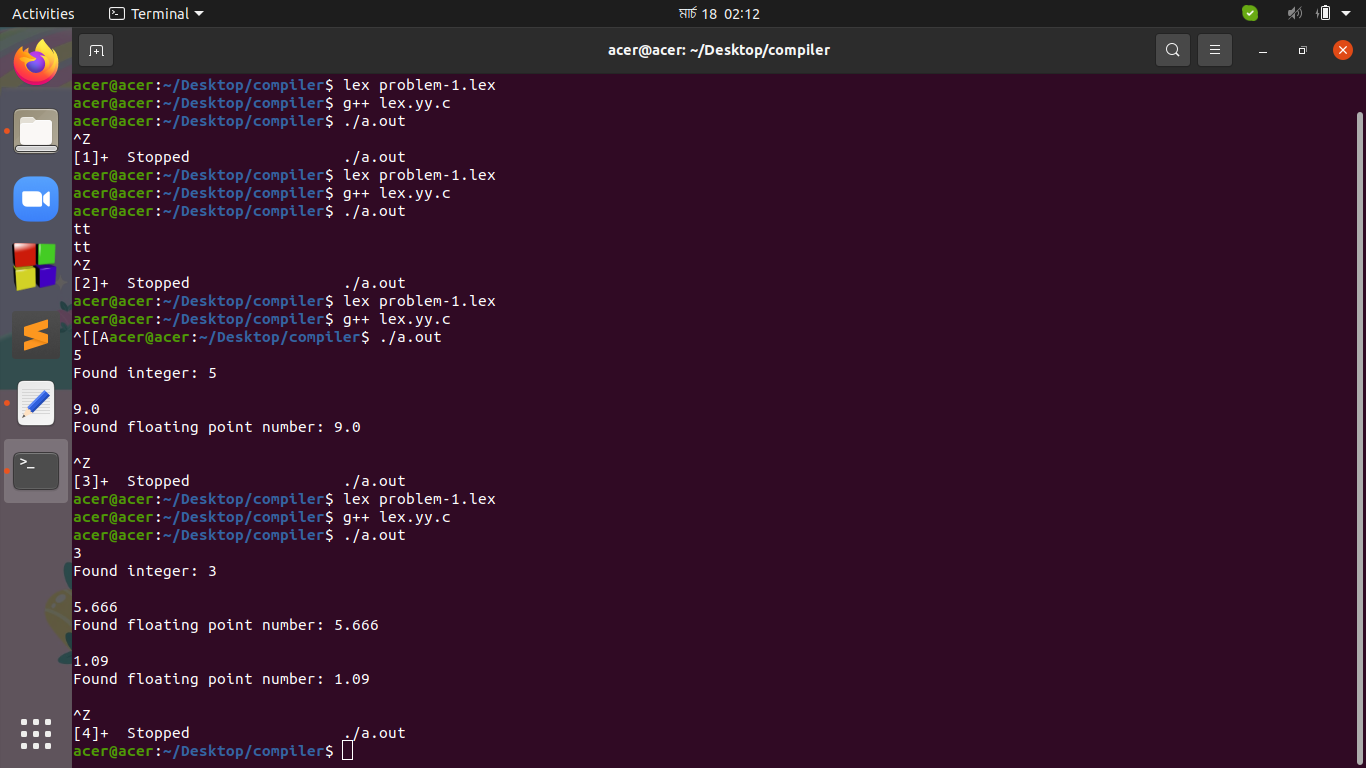
getchar();

getchar();

return 0;

}

**Output:**



**Problem-2**

In this assignment, your job is to program a simple lexical analyzer which recognizes all Arithmetic Operators, Relational Operators, and Logical Operators.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

%}

arithmetic\_operator "+"|"-"|"\*"|"/"|"%"|"++"|"--"

relational\_operator "=="|"!="|">"|"<"|">="|"<="

logical\_operator "&&"|"||"|"!"

%%

{arithmetic\_operator} {cout<<"Found arithmetic operator: "<<yytext<<endl;}

{relational\_operator} {cout<<"Found relational operator: "<<yytext<<endl;}

{logical\_operator} {cout<<"Found logical operator: "<<yytext<<endl;}

%%

int main()

{

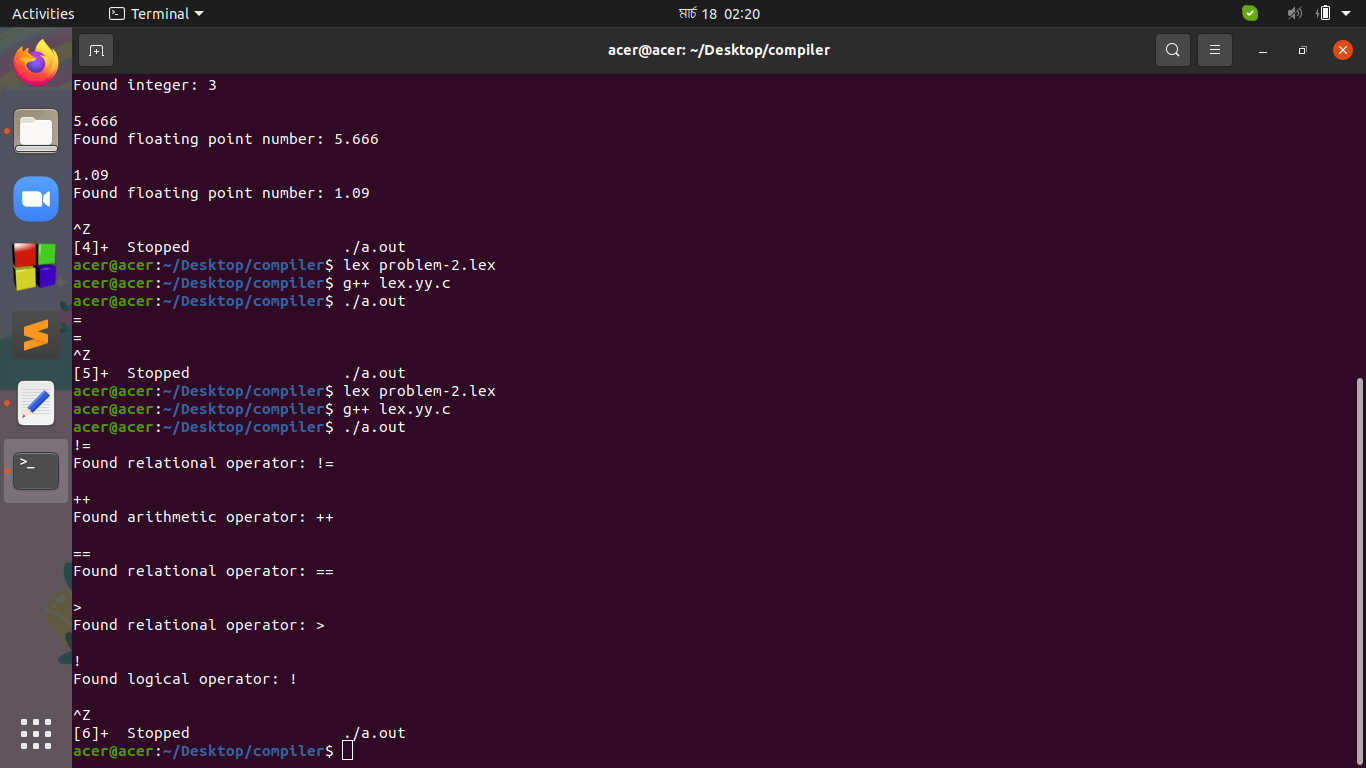
yylex();

getchar();

return 0;

}

**Output:**



**Problem-3**

In this assignment, your job is to program a simple lexical analyzer which recognizes all special characters and count number of uppercase and lowercase characters.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

int countUpper=0;

int countLower=0;

%}

special\_character "\n"|"\t"|"\v"|"\b"|"\r"|"\f"|"\a"|"\\"|"\?"|"\'"|"\""|"\ooo"|"\xhhh"|"\0"

uppercase\_character [A-Z]

lowercase\_character [a-z]

%%

{special\_character} {cout<<"Found special character"<<endl;}

{uppercase\_character} {cout<<"Found uppercase character: "<<yytext<<endl; countUpper++;}

{lowercase\_character} {cout<<"Found lowercase character: "<<yytext<<endl; countLower++;}

%%

int main()

{

yylex();

cout<<"Total number of uppercase characters: "<<countUpper<<endl;

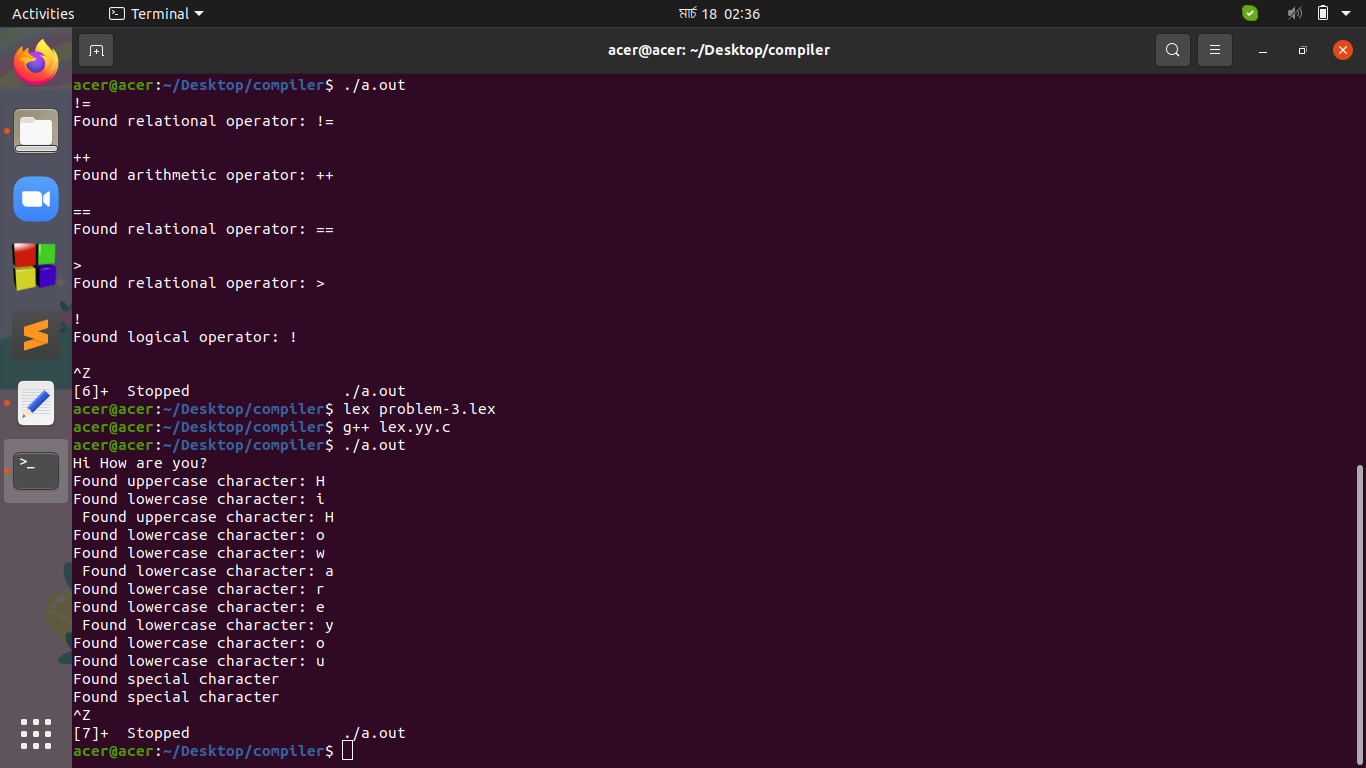
cout<<"Total number of lowercase characters: "<<countLower<<endl;

getchar();

return 0;

}

**Output:**



**Problem-4**

In this assignment, your job is to program a simple lexical analyzer which recognizes decimal and hexadecimal numbers.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

int countHex=0;

int countDec=0;

%}

dec [0-9]

hex [0-9A-F]

%%

{dec}+ {cout<<"Found decimal number: "<<yytext<<endl; countDec++;}

{hex}+ {cout<<"Found hexadecimal number: "<<yytext<<endl; countHex++;}

%%

int main()

{

yylex();

cout<<"Total number of decimal numbers: "<<countDec<<endl;

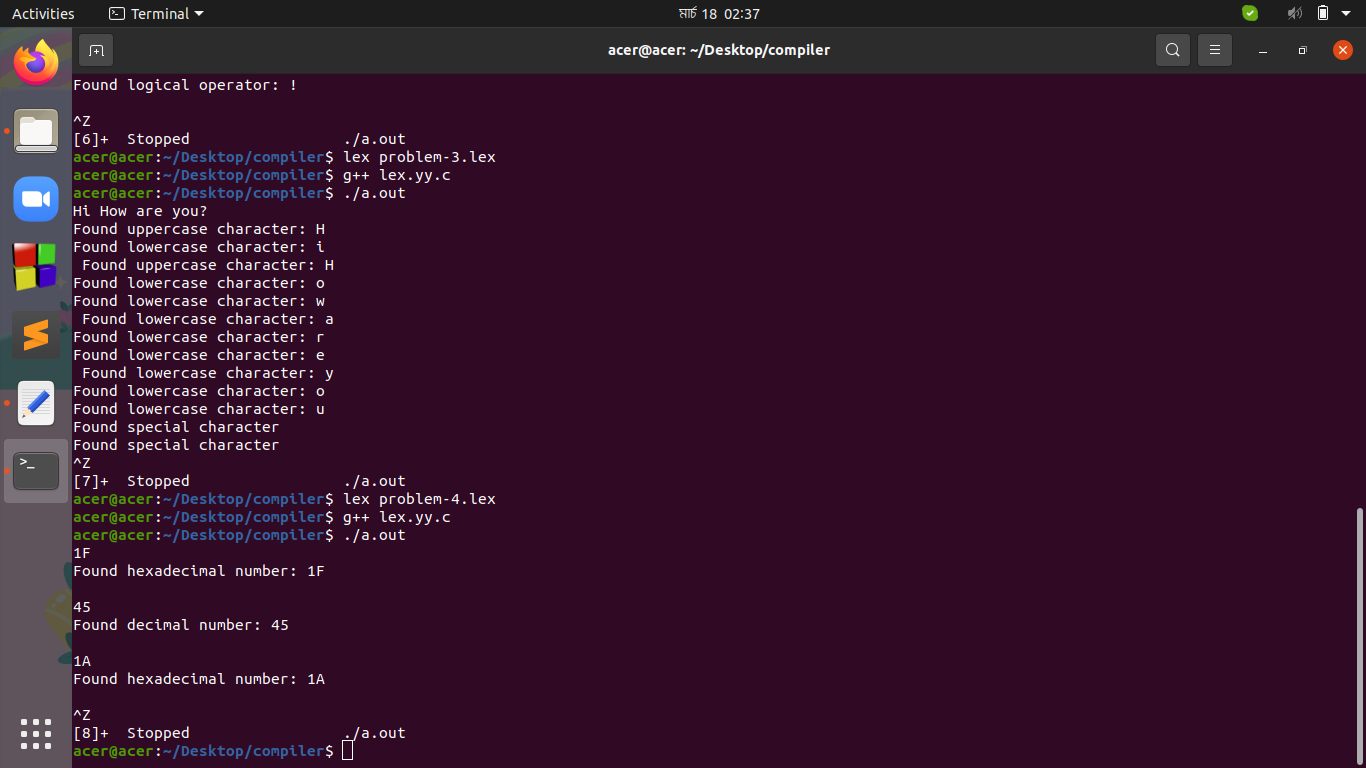
cout<<"Total number of hexdecimal numbers: "<<countHex<<endl;

getchar();

return 0;

}

**Output:**



**Problem-5**

In this assignment, your job is to program a simple lexical analyzer which recognizes characters, words and lines. Then print the number of characters, number of lines and the number of words in the input.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

int countChar=0;

int countWord=0;

int countLine=0;

%}

line "\n"

char [0-9a-zA-Z]

%%

{line}+ {cout<<"Found a line"<<endl; countLine++;}

{char}" " {cout<<"Found a word"<<endl; countWord++; countChar++;}

{char}{line} {cout<<"Found a character. Found a word. Found a line. "<<endl; countChar++; countWord++; countLine++;}

{char} {cout<<"Found a character"<<endl; countChar++;}

. {cout<<"Found a character"<<endl; countChar++;}

%%

int main()

{

yylex();

cout<<"Total number of lines: "<<countLine<<endl;

cout<<"Total number of words: "<<countWord<<endl;

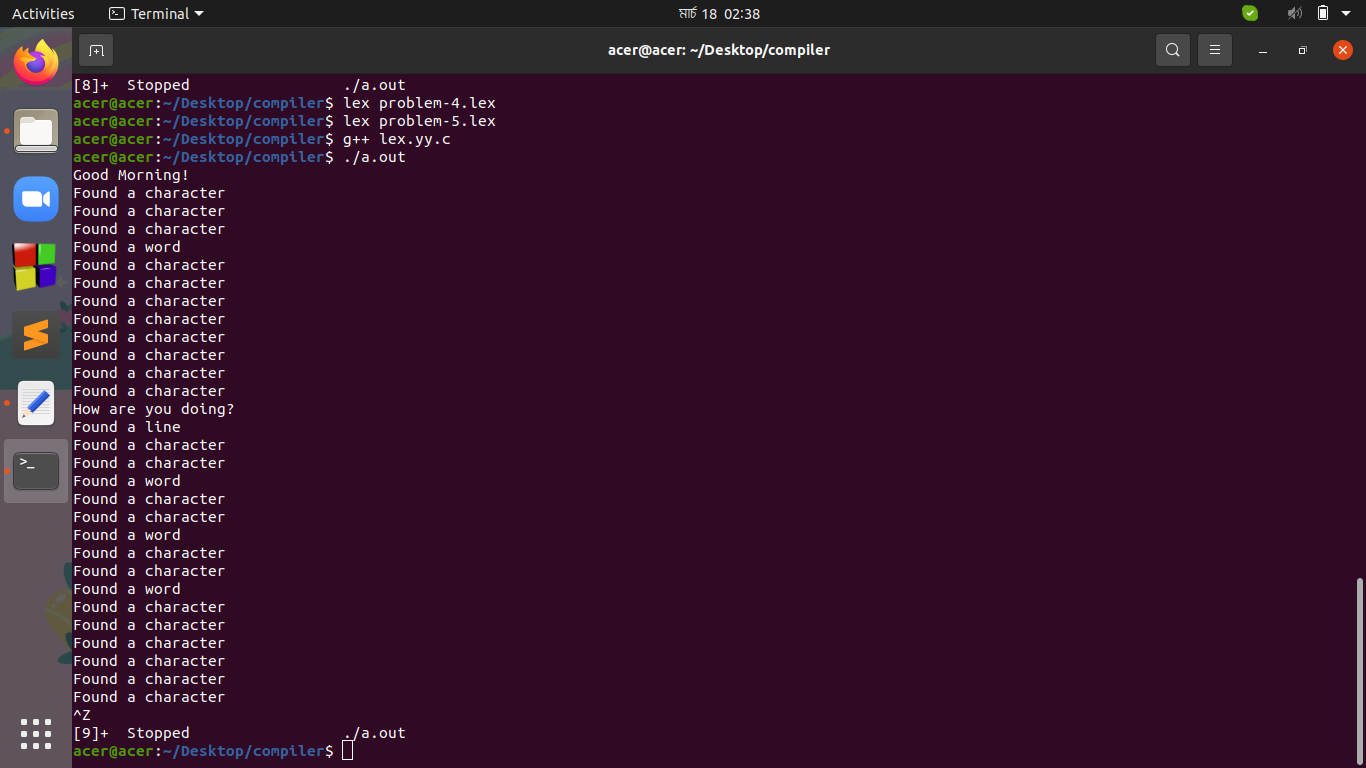
cout<<"Total number of characters: "<<countChar<<endl;

getchar();

return 0;

}

**Output:**



**Problem-6**

List out all C-like comments (both single line and multiline comments) from a text file.

**Solution Program:**

%option noyywrap

%{

#include<iostream>

using namespace std;

int sline=0,mline=0;

%}

%%

"//".\*"\n" {cout<<"Single line comment found. "<<yytext<<endl; sline++;}

"/\*".\*"\n"\*.\*"\*/" {cout<<"Multi line comment found. "<<yytext<<endl; mline++;}

%%

int main()

{

yyin=fopen("myfile.txt","r");

yyout=fopen("yourfile.txt","w");

yylex();

cout<<"Total number of single line comment: "<<sline<<endl<<"Total number of multi line comment: "<<mline<<endl;

fclose(yyin);

fclose(yyout);

getchar();

return 0;

}

**Input:**

From “myfile.txt”-

#include <stdio.h>:

int main(void){

float num; // declaring a float variable

/\* We can assign value to our variable like this \*/

num=17.873;

printf("num = %5.3f", num);

/\* this is a part of

c tutorial basic section\*/

return 0;

}

**Output:**

From “yourfile.txt”-

#include <stdio.h>:

int main(void){

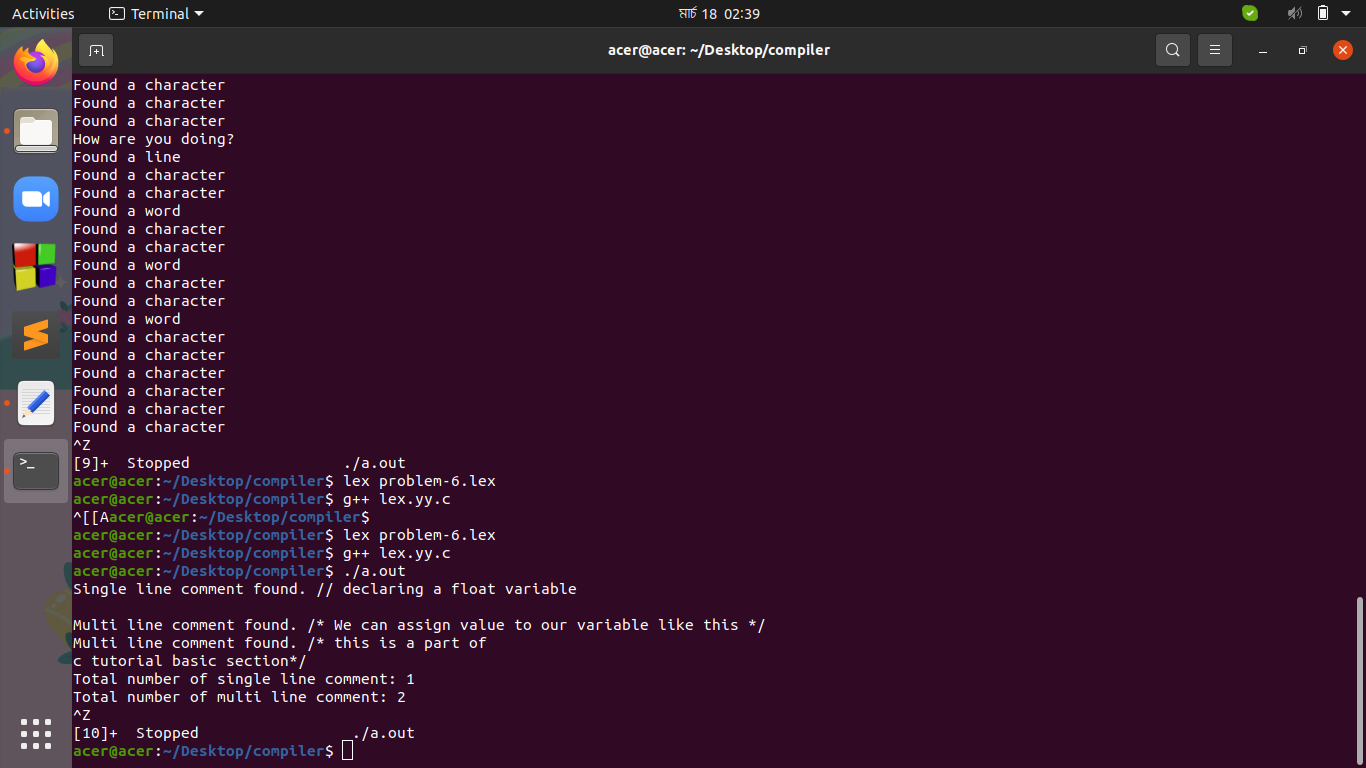
float num;

num=17.873;

printf("num = %5.3f", num);

return 0;

}



**Compiler Design Assignment using Flex**

**Problem Description:**

In this assignment, your job is to program a simple lexical analyzer that will build a symbol table from given stream of chars. You will need to read a file named "input.txt" to collect all chars. For simplicity, input file will be a C program without headers and methods. Then you will identify all the numerical values, identifiers, keywords, math operators, logical operators and others [distinct]. See the example for more details. You can assume that, there will be a space after each keywords. But, removal of space will add bonus point.

**Solution Program:**

%option noyywrap

%{

#include<bits/stdc++.h>

using namespace std;

string key\_string="";

string iden\_string="";

string math\_string="";

string rel\_string="";

string logi\_string="";

string num\_string="";

string oth\_string="";

int key=0,iden=0,math=0,rel=0,logi=0,num=0,oth=0;

void key\_fun(string st);

void iden\_fun(string st);

void math\_fun(string st);

void rel\_fun(string st);

void logi\_fun(string st);

void num\_fun(string st);

void oth\_fun(string st);

%}

keyword "printf"|"scanf"|"auto"|"double"|"int"|"struct"|"break"|"else"|"long"|"switch"|"case"|"enum"|"register"|"typedef"|"char"|"extern"|"return"|"union"|"continue"|"for"|"signed"|"void"|"do"|"if"|"static"|"while"|"default"|"goto"|"sizeof"|"volatile"|"const"|"float"|"short"|"unsigned"

letter [a-zA-Z]

digit [0-9]

math\_operator "+"|"-"|"\*"|"/"|"%"|"++"|"--"|"="

relational\_operator "=="|"!="|">"|"<"|">="|"<="

logical\_operator "&&"|"||"|"!"

delim [ \t\n]

ws {delim}+

%%

{ws} ;

{keyword} {key++; key\_fun(yytext);}

"\_"\*{letter}+{digit}\* {iden++; iden\_fun(yytext);}

{math\_operator} {math++; math\_fun(yytext);}

{relational\_operator} {rel++; rel\_fun(yytext);}

{logical\_operator} {logi++; logi\_fun(yytext);}

{digit}+"."\*{digit}\* {num++; num\_fun(yytext);}

. {oth++; oth\_fun(yytext);}

%%

int main()

{

yyin=fopen("input.txt", "r");

yyout=fopen("output.txt", "w");

yylex();

char\* ara1=&key\_string[0];

char\* ara2=&iden\_string[0];

char\* ara3=&math\_string[0];

char\* ara4=&rel\_string[0];

char\* ara5=&logi\_string[0];

char\* ara6=&num\_string[0];

char\* ara7=&oth\_string[0];

if(key!=0) fprintf(yyout,"Keywords: %s\n",ara1);

if(iden!=0) fprintf(yyout,"Identifiers: %s\n",ara2);

if(math!=0) fprintf(yyout,"Math Operators: %s\n",ara3);

if(rel!=0) fprintf(yyout,"Relational Operators: %s\n",ara4);

if(logi!=0) fprintf(yyout,"Logical Operators: %s\n",ara5);

if(num!=0) fprintf(yyout,"Numerical Values: %s\n",ara6);

if(oth!=0) fprintf(yyout,"Others: %s\n",ara7);

fclose(yyin);

fclose(yyout);

getchar();

return 0;

}

void key\_fun(string st)

{

if(key>1) {if(key\_string.find(st)==string::npos) key\_string=key\_string+", "+st;}

else if(key==1) key\_string=key\_string+st;

}

void iden\_fun(string st)

{

if(iden>1) {if(iden\_string.find(st)==string::npos) iden\_string=iden\_string+", "+st;}

else if(iden==1) iden\_string=iden\_string+st;

}

void math\_fun(string st)

{

if(math>1) {if(math\_string.find(st)==string::npos) math\_string=math\_string+", "+st;}

else if(math==1) math\_string=math\_string+st;

}

void rel\_fun(string st)

{

if(rel>1) {if(rel\_string.find(st)==string::npos) rel\_string=rel\_string+", "+st;}

else if(rel==1) rel\_string=rel\_string+st;

}

void logi\_fun(string st)

{

if(logi>1) {if(logi\_string.find(st)==string::npos) logi\_string=logi\_string+", "+st;}

else if(logi==1) logi\_string=logi\_string+st;

}

void num\_fun(string st)

{

if(num>1) {if(num\_string.find(st)==string::npos) num\_string=num\_string+", "+st;}

else if(num==1) num\_string=num\_string+st;

}

void oth\_fun(string st)

{

if(oth>1) {if(oth\_string.find(st)==string::npos) oth\_string=oth\_string+" "+st;}

else if(oth==1) oth\_string=oth\_string+st;

}

**Input:**

From “input.txt”-

int a, b, c;

float d, e;

a = b = 5;

c = 6;

if ( a > b)

{

c = a - b;

e = d - 2.0;

for(int f=5; f<=0; f++)

{

printf(“%d”,f);

}

}

else

{

d = e + 6.0;

b = a + c;

}

**Output:**

From “output.txt”-

Keywords: int, float, if, for, printf, else

Identifiers: a, b, c, d, e, f

Math Operators: =, -, ++, %

Relational Operators: >, <=

Numerical Values: 5, 6, 2.0, 6.0

Others: , ; ( ) { â € œ  }

