CD Assignment 2

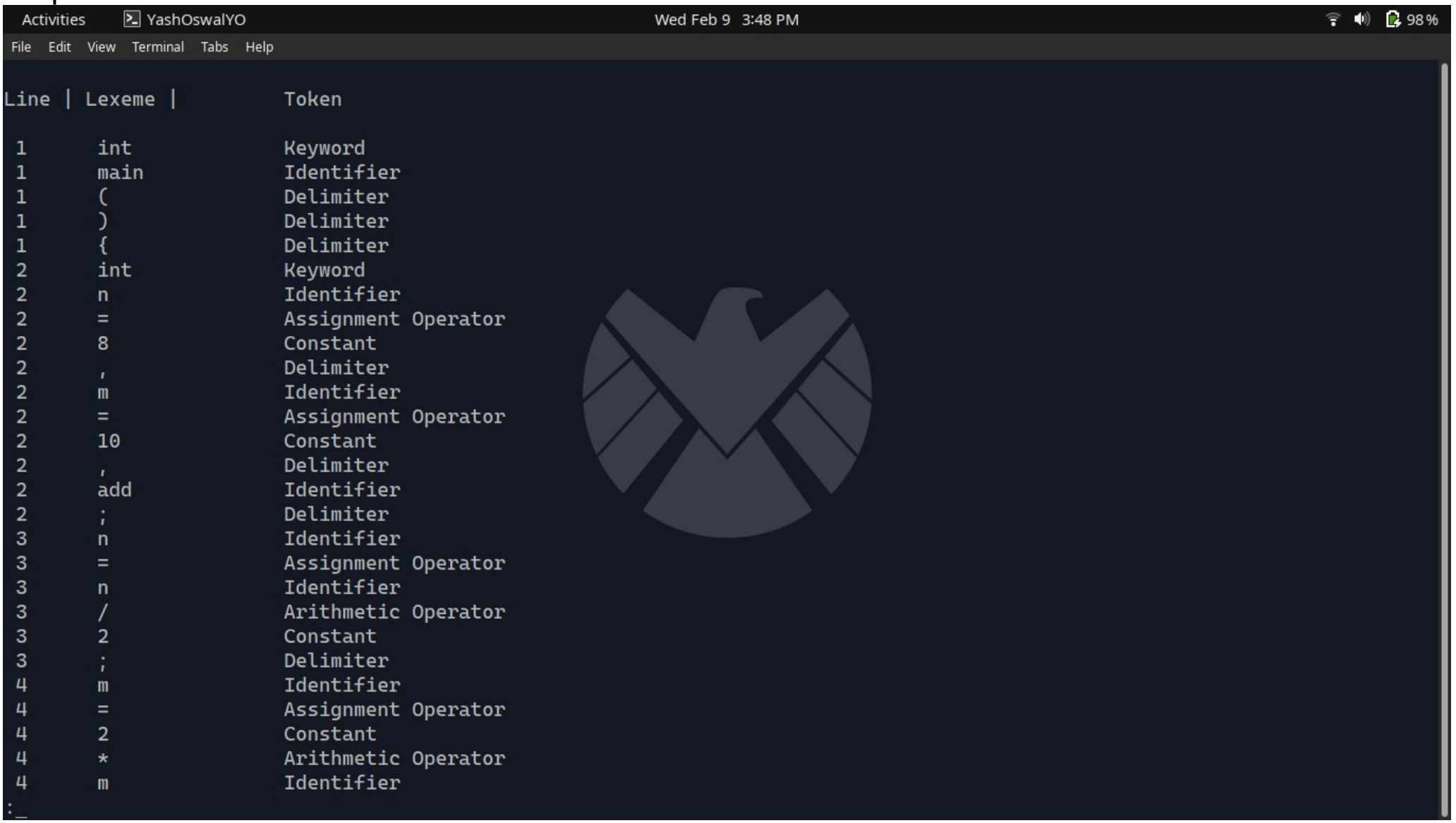
Name: Yash Oswal
Div: B Roll no.: 38
SRN: 201901226

Design a Lexical analyzer for the subset of C Language using LEX or FLEX. Read input from file. Also Create symbol table. Upload single file with input , output and source code.

Input Code:- input.txt

```
int main(){
int n=8, m=10, add;
n=n/2;
m=2*m;
add=n+m;
return 0;
}
```

Output:


Activities
YashOswalYO
Wed Feb 9 3:48 PM
98 %
File Edit View Terminal Tabs Help
Line | Lexeme | Token
1 int Keyword
1 main Identifier
1 (Delimiter
1) Delimiter
1 { Delimiter
2 int Keyword
2 n Identifier
2 = Assignment Operator
2 8 Constant
2 , Delimiter
2 m Identifier
2 = Assignment Operator
2 10 Constant
2 , Delimiter
2 add Identifier
2 ; Delimiter
3 n Identifier
3 = Assignment Operator
3 n Identifier
3 / Arithmetic Operator
3 2 Constant
3 ; Delimiter
4 m Identifier
4 = Assignment Operator
4 2 Constant
4 * Arithmetic Operator
4 m Identifier
:_

```
 4      =               Assignment Operator
 4      2               Constant
 4      *               Arithmetic Operator
 4      m               Identifier
 4      ;               Delimiter
 5      add             Identifier
 5      =               Assignment Operator
 5      n               Identifier
 5      +               Arithmetic Operator
 5      m               Identifier
 5      ;               Delimiter
 6      return          Keyword
 6      0               Constant
 6      ;               Delimiter
 7      }               Delimiter


Symbol Table :
-------------------------
| Line  |       Lexeme  |
-------------------------
| 1     |       main    |
| 2     |       n       |
| 3     |       m       |
| 4     |       add     |
-------------------------
(END)
```

Source Code :-

```
%{
#include <stdio.h>
#include<string.h>
struct symEntry{
 int index;
 char lexeme[30];
};
struct symEntry symtable[30];
int sti=0;
int line=1;

void put_symtab(){
 int j;
 if(sti==0){
 symtable[sti].index=sti+1;
 strcpy(symtable[sti].lexeme,yytext);
 sti++;
 return;
 }
 for(j=0;j<sti;j++){
 if(strcmp(symtable[j].lexeme,yytext)==0){
 return;
 }
 }
 symtable[sti].index=sti+1;
 strcpy(symtable[sti].lexeme,yytext);
 sti++;
}
%}
letter [a-zA-Z]
number [0-9]
delim ["|']
%%
```

```
"int"|"if"|"double"|"long"|"goto"|"static"|"float"|"short"|"while"|"char"|"const"|"void"|"else"|"return"|"
printf"|"scanf" {printf("\n %d\t%s \t\tKeyword", line,yytext);}
"("|")"|"{"|"}"|"["|"]"|";"|"," {printf("\n %d\t%s \t\tDelimiter", line, yytext);}




{delim} {printf("\n %d\t%s \t\tDelimiter", line, yytext);}
"+"|"-"|"*"|"%"|"/"|"++"|"--" {printf("\n %d\t%s \t\tArithmetic Operator",line, yytext);}
"=="|"<"|">"|"<="|">=" {printf("\n %d\t%s \t\tRelational Operator",line, yytext);}
"=" {printf("\n %d\t%s \t\tAssignment Operator",line, yytext);}
{letter}+|({letter}{number})* {printf("\n %d\t%s \t\tIdentifier",line, yytext);put_symtab();}
{number}+ {printf("\n %d\t%s\t\tConstant",line, yytext);}
{number}+{letter}+ {printf("\n %d\t%s \tERROR This is ILLEGAL",line,yytext);}
{delim}({letter}|{number})*{delim} {printf("\n %d\t%s \tString Constant/Literal",line,yytext);}
"\n" {line++;}
%%



void print_st(){
 int j;
 printf("\nSymbol Table : \n");
 printf("-------------------------\n");
 printf("| Line\t|\tLexeme\t|\n");


printf("-------------------------\n");
 for(j=0;j<sti;j++){
 printf("| %d\t|\t%s\t|\n",symtable[j].index,symtable[j].lexeme);
 }
 printf("-------------------------\n");
}
int main(int argc, char* argv[])
{
yyin = fopen(argv[1], "r");
printf("\nLine | Lexeme | \tToken\n");
```

```
yylex();
 printf("\n\n");
 print_st();
fclose(yyin);
}
int yywrap()
{
return 1;
}
```