Practical	1
Name	Niketkumar Kothari
Roll No.	18BCE134
Subject	Complier Construction
AIM	To implement lexical analyse to recognize all distinct token classes.

Definition :

Source Language: C

Target Language: Assembly-TASM

Data Types: int, float, char

Control construct: if else

Loop construct: for-while

Operators: (+,-,*,/,=,>,<,>=,<=) logical

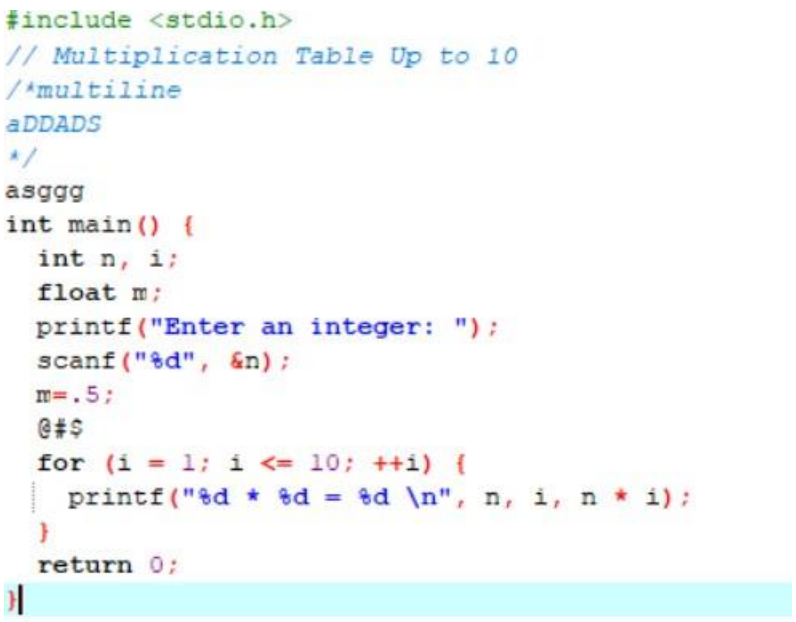
CODE:

```
%{

//Lex Program to recognize C tokens

int l=1; int count(char str[]);

%}

operator (\+)|(\-)|(\*)|(\/)|(\^)|(\&)|(\&\&)|(\|)|(\|\|)|(=)|(==) ternary

(.*)\?(.*):

digit [0-9]+

alphanumeric [a-zA-Z_][a-zA-Z0-9_]*

fdigit [0-9]*"."[0-9]+  char \'([a-zA-

Z])\'  string \".*\"

%%

"//".*                                             {}

"/*"(.|"\n")*"*/"                                                    {l+=count(yytext);}

\n            l++; {printf("\n",yytext);}

"if"|"else"|"for"|"while"|"main"|"return"   {printf("%s ",yytext);}

"int"|"float"|"char"                         {printf("DT ",yytext);}
{digit}                                                    {printf("NUM ",yytext);}
```

```
{alphanumeric}                                          {printf("IDENTIFIER
",yytext);}
{operator}                 {printf("%s ",yytext);}
{string}                                          {printf("STR ",yytext);}
{fdigit}                              {printf("FLOAT ",yytext);}
{char}                                {printf("CHAR ",yytext);}
[(),;{}]                              {printf("%s ",yytext);}
#include.*                                        {printf("Include File",yytext);}
#define.*            {printf("Macro Definition",yytext);}
" "|"\t"                          {printf(" ");}
.                                             {printf("Invalid lexeme at %d",l);}


%%
//Driver Code to read source code and return tokens
int main() {     yylex();

        return 0;
}
int count (char str[])
{  int c =0; int
i=0;
while(str[i]!='\0')

{
        if(str[i]=='\n')
        c++; i++; }
return c;

}
int yywrap()
{ return
1;

}
```

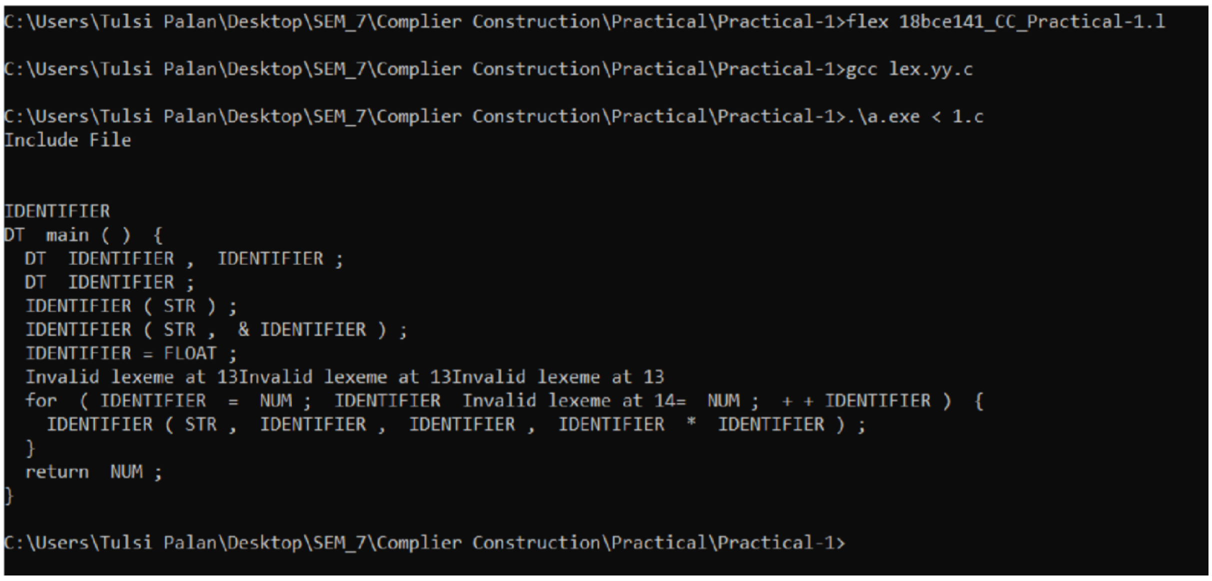
OUTPUT:

1.c file



```c
#include <stdio.h>
// Multiplication Table Up to 10
/*multiline
aDDADS
*/
asggg
int main() {
  int n, i;
  float m;
  printf("Enter an integer: ");
  scanf("%d", &n);
  m=.5;
  @#$
  for (i = 1; i <= 10; ++i) {
    printf("%d * %d = %d \n", n, i, n * i);
  }
  return 0;
}
```

Output



```
C:\Users\Tulsi Palan\Desktop\SEM_7\Complier Construction\Practical\Practical-1>flex 18bce141_CC_Practical-1.l

C:\Users\Tulsi Palan\Desktop\SEM_7\Complier Construction\Practical\Practical-1>gcc lex.yy.c

C:\Users\Tulsi Palan\Desktop\SEM_7\Complier Construction\Practical\Practical-1>.\a.exe < 1.c
Include File


IDENTIFIER
DT  main ( )  {
  DT  IDENTIFIER ,  IDENTIFIER ;
  DT  IDENTIFIER ;
  IDENTIFIER ( STR ) ;
  IDENTIFIER ( STR ,  & IDENTIFIER ) ;
  IDENTIFIER = FLOAT ;
  Invalid lexeme at 13Invalid lexeme at 13Invalid lexeme at 13
  for  ( IDENTIFIER  =  NUM ;  IDENTIFIER  Invalid lexeme at 14=  NUM ;  + + IDENTIFIER )  {
    IDENTIFIER ( STR ,  IDENTIFIER ,  IDENTIFIER ,  IDENTIFIER  *  IDENTIFIER ) ;
  }
  return  NUM ;
}

C:\Users\Tulsi Palan\Desktop\SEM_7\Complier Construction\Practical\Practical-1>
```