Program:

```
//Dasarada Ram Reddy - 160114733092

/* Program for Tokenization (counting the no of characters, lines, spaces, words,
tabs, integer, float, Sum of the given integer & float etc..,).*/
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
#include<math.h>
int l,w,i,f,ch;
bool isInt(char *,int *);
bool isFloat(char *,float *);
int main()
{
   FILE *fp;
   l=w=i=f=ch=0;
   char c;
   int sumi=0;
   float sumf=0.0;
   fp=fopen("data.txt","r");char buf[100];int p=0;
   if(fp==NULL)
   exit(1);
   else
   {
      while((c=fgetc(fp))!=EOF)
      {
         if(c!=' '&&c!='\n'&&c!='\r')
         {
            buf[p++]=c;
                ch++;
         }
         else // if(c==' '||c=='\n'||c=='\r')
         {
            buf[p]='\0';
            //printf("%s\n",buf);
            if(c=='\n')//||c=='\r')
            l++;
            int getint;float getfloat;
            if(isInt(buf,&getint))
            {
                        printf("integer:%s\n",buf);
          //      i++;
```

```
                sumi+=getint;
              }
              else if(isFloat(buf,&getfloat))
              {
                         printf("float:%s\n",buf);
           //       f++;
                sumf+=getfloat;
              }
                else
                {
                   printf("word:%s\n",buf);
                w++;
                }
              buf[p]='\0';
              p=0;
           }
          // printf("%d %d %d %d %s",i,l,f,w,buf);

        }
     }
printf("sum if ints %d\n",sumi);
printf("sum of floats %f\n",sumf);
printf("no of words %d\n",w);
printf("no of lines %d\n",l);
printf("no of integers %d\n",i);
printf("no of floats %d\n",f);
printf("no of characters %d\n",ch);
}
bool isInt(char *s,int *int_val)
{
  //    printf("hi %d\n",*getint);
        *int_val=0;
   int sign=1;
   int p;
   if(s[0]=='-')
      sign=-1;
   if(s[0]=='+'||s[0]=='-'||(s[0]>=48&&s[0]<=57))
   {
      if(s[0]>=48&&s[0]<=57)
      *int_val=(int)(s[0]-48);
   }
   else
      return false;
```

```
    for(p=1;s[p]!='\0';p++)
    {
       if(s[p]>=48&&s[p]<=57)
       {
          *int_val=(*int_val)*10+(int)(s[p]-48);
       }
       else
          return false;

    }
    *int_val=(*int_val)*sign;
         i++;
    return true;
 }
 bool isFloat(char *s,float *float_val)
 {
         //printf("hi 3 : %s",s);
    float sign=1.0;
    *float_val=0.0;
    int i;int count=0;
    if(s[0]=='-')
       sign=-1.0;
    if(s[0]=='+'||s[0]=='-'||(s[0]>=48&&s[0]<=57))
    {
       if(s[0]>=48&&s[0]<=57)
       *float_val=(*float_val)*10.0+(float)(s[0]-48);
    }
    else
       return false;
         int flag=0;int power1=-1;
         //printf("hi 4:%f",*getfloat);
    for(i=1;s[i]!='\0';i++)
    {
       if(s[i]=='.'&&count==0)
       {
          count++;flag=1;
                continue;
       }
       else if(s[i]=='.'&&count>0)
          return false;
       if((s[i]>=48&&s[i]<=57)&&flag==0)
       {
          *float_val=(*float_val)*10+(s[i]-48);
```

```
        }
        else if((s[i]>=48&&s[i]<=57)&&flag==1)
        {
              *float_val=(*float_val)+(s[i]-48)*(pow(10,power1));
              power1--;
        }
        else
           return false;

    }
    *float_val=(*float_val)*sign;
         f++;
    return true;
}
```

Testing:

Input:

data.txt

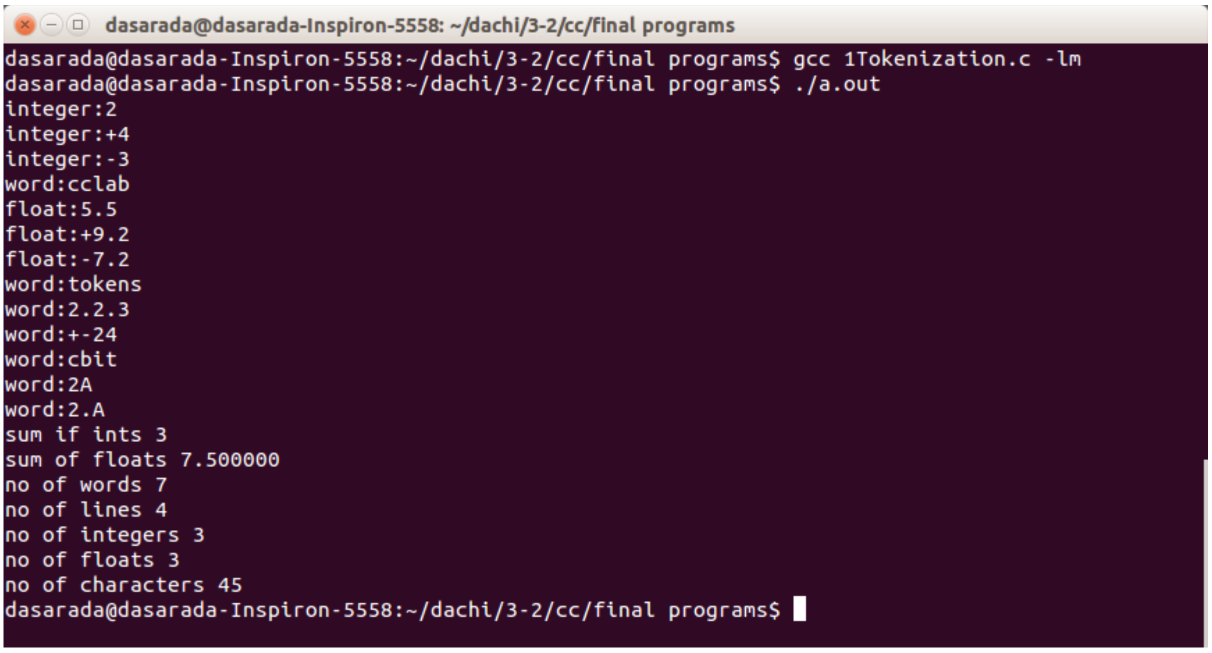
```
2 +4 -3 cclab
5.5 +9.2 -7.2 tokens
2.2.3 +-24 cbit
2A 2.A
```

Expected Output:

```
integer:2
integer:+4
integer:-3
word:cclab
float:5.5
float:+9.2
float:-7.2
word:tokens
word:2.2.3
word:+-24
word:cbit
word:2A
word:2.A
```

sum if ints 3
sum of floats 7.500000
no of words 7
no of lines 4
no of integers 3
no of floats 3
no of characters 45

Actual Output:



dasarada@dasarada-Inspiron-5558: ~/dachi/3-2/cc/final programs

```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/final programs$ gcc 1Tokenization.c -lm
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/final programs$ ./a.out
integer:2
integer:+4
integer:-3
word:cclab
float:5.5
float:+9.2
float:-7.2
word:tokens
word:2.2.3
word:+-24
word:cbit
word:2A
word:2.A
sum if ints 3
sum of floats 7.500000
no of words 7
no of lines 4
no of integers 3
no of floats 3
no of characters 45
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/final programs$
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy - 160114733092

// Program to implement Scanner using C.
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char key[32]
[10]={"auto","break","case","char","const","continue","default","do","double","
else","enum","extern","float","for","goto","if","int","long","register","return","s
hort","signed","sizeof","static","struct","switch","typedef","union","unsigned","
void","volatile","while"};
char echar[10]={'b','t','v','r','f','n','\\','?','0','a'},com[100];
int main()
{
 FILE *f;
 char c,x,y,z,temp[20],fr[5];
 int i,j,k,l=1,t=1,f1,flag,m,p;
 f=fopen("file.txt","r");
 if(f)
 {
  printf("Line\tToken no\tToken name\t\tLexeme\n");
  while((c=getc(f))!=EOF)
  {
   j=0;
   if(isalpha(c))
   {
        flag=0;
        temp[j++]=c;c=getc(f);
        while(isalnum(c))
        {
         temp[j++]=c;c=getc(f);
        }
        for(i=0;i<32;i++)
        {
         if(strcmp(temp,key[i])==0)
         {
          flag=1;
          break;
         }
        }
        if(flag==1)
        {
         printf("%d\t%d\tkeyword\t\t\t%s\n",l,t,temp);
         t++;
        }
        else
        {
         if(c=='.')
         {
          temp[j++]=c;
          for(p=0;p<5;p++)
          { fr[p]='\0'; }
          y=getc(f);
          while(y!='>')
          {
           fr[p++]=y;
           temp[j++]=y;
           y=getc(f);
           if(y==';'||y=='"')  break;
          }
          fseek(f,-(j+10),SEEK_CUR);
          y=getc(f);
          fseek(f,(j+9),SEEK_CUR);
          if(y=='#')
          {
           printf("%d\t%d\theader\t\t\t%s\n",l,t,temp);
          }
          else
          { printf("%d\t%d\tidentifier\t\t%s\n",l,t,temp); t++;
          fseek(f,-1,SEEK_CUR);}
         }
         else
         {
          fseek(f,-1,SEEK_CUR);
          printf("%d\t%d\tidentifier\t\t%s\n",l,t,temp);
           t++;
         }
        }
}
else if(c=='+'||c=='='||c=='-'||c=='<'||c=='>'||c=='*'||c=='/')
{
        if(c=='+')
        {
         x=getc(f);
        if(x=='+')
        {
         temp[j++]=c;
         temp[j++]=x;
         printf("%d\t%d\tincrementer\t\t%s\n",l,t,temp);
         t++;
 }
}
else if(c=='-')
{
        x=getc(f);
        if(x=='-')
        {
         temp[j++]=c;
         temp[j++]=x;
         printf("%d\t%d\tdecrementer\t\t%s\n",l,t,temp);
         t++;
 }
}
else if(c=='/')
{
        x=getc(f);
        if(x=='*')
        {
         temp[j++]=c;
         temp[j++]=x;
         printf("%d\t%d\tcomment starts\t\t%s\n",l,t,temp);
         t++;
         x=getc(f);
         k=0;
         while(x!='*')
         {
                 if(x=='\n')
                  x=' ';
                 com[k++]=x;
                 x=getc(f);
         }
         fseek(f,-1,SEEK_CUR);
         printf("%d\t%d\tcomment\t\t\t%s\n",l,t,com);
         t++;
         for(k=0;k<100;k++)
          com[k]='\0';
        }
        if(x=='/')
        {
         temp[j++]=c;
         temp[j++]=x;
         printf("%d\t%d\tcomment starts\t\t%s\n",l,t,temp);
         t++;
         x=getc(f);
         k=0;
         while(x!='\n')
         {
                com[k++]=x;
                x=getc(f);
         }
         fseek(f,-1,SEEK_CUR);
         printf("%d\t%d\tcomment\t\t\t%s\n",l,t,com);
         t++;
         for(k=0;k<100;k++)
          com[k]='\0';
        }
 }
 else if(c=='*')
 {
        x=getc(f);
        if(x=='/')
        {
         temp[j++]=c;
         temp[j++]=x;
         printf("%d\t%d\tcomment ends\t\t%s\n",l,t,temp);
         t++;
        }
 }
 else
 {
        printf("%d\t%d\toperator\t\t%c\n",l,t,c);
        t++;
  }
}
else if(isdigit(c))
{
       temp[j++]=c;
       c=getc(f);
       while(isdigit(c))
       {
          temp[j++]=c;
          c=getc(f);
         }
         fseek(f,-1,SEEK_CUR);
         printf("%d\t%d\tdigit\t\t\t%s\n",l,t,temp);
         t++;
  }
  else if(c=='{'||c=='}'||c==';'||c==','||c=='('||c==')'||c=='?'||c=='!')
  {
         printf("%d\t%d\tspecial symbol\t\t%c\n",l,t,c);t++;
  }
  else if(c=='#')
  {
         printf("%d\t%d\tpreprocesssor\t\t%c\n",l,t,c);t++;
  }
  else if(c=='\\')
  {
         y=getc(f);
         for(i=0;i<3;i++)
         {
          if(y==echar[i])
          {
           temp[j++]=c;
           temp[j++]=y;
           printf("%d\t%d\t\tesacpe character\t%s\n",l,t,temp);
           t++;
          }
         }
  }
  else if(c==' '){ }
  else if(c=='\n'){l++;}
  else{ }
  for(i=0;i<20;i++)
          temp[i]='\0';
 }
}
else
{
 printf("\nfp file doesnot exist\n");
 return -1;
}
fclose(f);
return 0;
```

```
}
```

Testing:

Input:

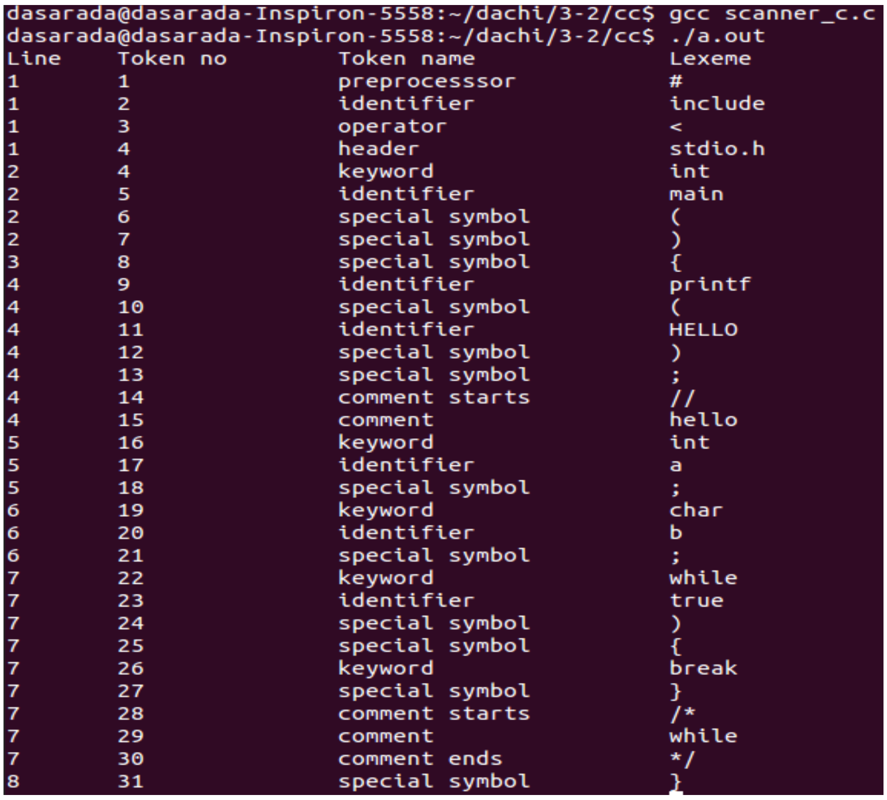
```
//input.c

#include<stdio.h>
int main()
{
	printf("HELLO");//hello
	int a;
	char b;
	while(true){break;}/*while*/
}
```

Expected Output:

```
Line  Token no    Token name        Lexeme
1     1           preprocesssor     #
1     2           identifier        include
1     3           operator          <
1     4           header            stdio.h
2     4           keyword           int
2     5           identifier        main
2     6           special symbol    (
2     7           special symbol    )
3     8           special symbol    {
4     9           identifier        printf
4     10          special symbol    (
4     11          identifier        HELLO
4     12          special symbol    )
4     13          special symbol    ;
4     14          comment starts    //
4     15          comment           hello
5     16          keyword           int
5     17          identifier        a
5     18          special symbol    ;
6     19          keyword           char
6     20          identifier        b
6     21          special symbol    ;
7     22          keyword           while
7     23          identifier        true
7     24          special symbol    )
```

7	25	special symbol	{
7	26	keyword	break
7	27	special symbol	}
7	28	comment starts	/*
7	29	comment	while
7	30	comment ends	*/
8	31	special symbol	}

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ gcc scanner_c.c
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Line    Token no        Token name              Lexeme
1       1               preprocesssor           #
1       2               identifier              include
1       3               operator                <
1       4               header                  stdio.h
2       4               keyword                 int
2       5               identifier              main
2       6               special symbol          (
2       7               special symbol          )
3       8               special symbol          {
4       9               identifier              printf
4       10              special symbol          (
4       11              identifier              HELLO
4       12              special symbol          )
4       13              special symbol          ;
4       14              comment starts          //
4       15              comment                 hello
5       16              keyword                 int
5       17              identifier              a
5       18              special symbol          ;
6       19              keyword                 char
6       20              identifier              b
6       21              special symbol          ;
7       22              keyword                 while
7       23              identifier              true
7       24              special symbol          )
7       25              special symbol          {
7       26              keyword                 break
7       27              special symbol          }
7       28              comment starts          /*
7       29              comment                 while
7       30              comment ends            */
8       31              special symbol          }
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy - 160114733092

/* Program to implement Scanner application using LEX.*/
%{
#include<stdio.h>
#include<string.h>
int t=1;
int l=1,i;
%}

hfile assert.h|complex.h|ctype.h|errno.h|fenv.h|float.h|inttypes.h|iso646.h|
limits.h|locale.h|math.h|setjmp.h|signal.h|stdalign.h|stdarg.h|stdatomic.h|
stdbool.h|stddef.h|stdint.h|stdio.h|stdlib.h|stdnoreturn.h|string.h|tgmath.h|
threads.h|time.h|uchar.h|wchar.h|wctype.h
key 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
file txt|h|c
format c|d|e|E|f|g|G|o|s|u|x|X
func fopen|fclose|getchar|putchar|printf|scanf|strcat|strcmp|strcpy|isdigit|isalpha|
isalnum|islower|isupper|acos|asin|atan|cos|exp|fabs|sqrt|time|difftime|clock|
malloc|rand|srand
type int|char|double|float
id [a-zA-Z][a-zA-Z0-9_]*
size [0-9]+
index [a-zA-Z]
s {size}|{index}|{index}"++"|{index}--|"--"{index}|"++"{index}
log &&|"||"|!
num [0-9]+
str [a-zA-Z-_]
mode r|w|a|r+|w+|a+
ffile "\""{id}"\."{file}"\""

%%

{id}"="([+-]?{num})|("""{str}""")       {printf("\n%d\t%d\t\tdefinition\t
%s",l,t,yytext);t++;}
"*"[ ]?{id}            {printf("\n%d\t%d\t\tpointer\t\t%s",l,t,yytext);t++;}
"//"[^\n]+"\n"         {printf("\n%d\t%d\t\tcomment\t\t",l,t);t++;
                         for(i=2;i<yyleng-1;i++)
                          printf("%c",yytext[i]);
```

```
                                l++;}
[;]                     {printf("\n%d\t%d\t\tterminator\t%s",l,t,yytext);t++;}
{func}                  {printf("\n%d\t%d\t\tfunction\t%s",l,t,yytext);t++;}
"%"{format}             {printf("\n%d\t%d\t\tformat\t\t%s",l,t,yytext);t++;}
["$&^{}(),'#]           {printf("\n%d\t%d\t\tspecial char\t%s",l,t,yytext);t++;}
[+-=*/%]                {printf("\n%d\t%d\t\toperator\t%s",l,t,yytext);t++;}
{log}                   {printf("\n%d\t%d\t\tlogical op\t%s",l,t,yytext);t++;}
{key}                   {printf("\n%d\t%d\t\tkeyword\t\t%s",l,t,yytext);t++;}
{hfile}                 {printf("\n%d\t%d\t\theader file\t%s",l,t,yytext);t++;}
{id}                    {printf("\n%d\t%d\t\tidentifier\t%s",l,t,yytext);t++;}
[a-z]+"."[a-z]+         {printf("\n%d\t%d\t\tidentifier\t%s",l,t,yytext);t++;}
{id}"++"                {printf("\n%d\t%d\t\tincrementer\t%s",l,t,yytext);t++;}
{id}"--"                {printf("\n%d\t%d\t\tdecrementer\t%s",l,t,yytext);t++;}
"=="                    {printf("\n%d\t%d\t\tequality check\t%s",l,t,yytext);t++;}
"\\"[btvrfn?0a]         {printf("\n%d\t%d\t\tescape char\t%s",l,t,yytext);t++;}
[+-]?{num}              {printf("\n%d\t%d\t\tnumber\t\t%s",l,t,yytext);t++;}
{id}"["{s}"]"           {printf("\n%d\t%d\t\t1-D array\t%s",l,t,yytext);t++;}
"\""[a-zA-Z0-9]"\"" {printf("\n%d\t%d\t\tcharacter\t%s",l,t,yytext);t++;}
"\""[a-zA-Z0-9]+"\""{printf("\n%d\t%d\t\tstring\t\t%s",l,t,yytext);t++;}
{id}"["{s}"]""["{s}"]"
                        {printf("\n%d\t%d\t\t2-D array\t%s",l,t,yytext);t++;}
"+="|"-="|"*="|"/="|"%="
                        {printf("\n%d\t%d\t\tcompound op\t%s",l,t,yytext);t++;}
"<="|">="|"<"|">"|"!="
                        {printf("\n%d\t%d\t\trelational op\t%s",l,t,yytext);t++;}
"("{type}")"            {printf("\n%d\t%d\t\ttype cast to\t",l,t);t++;
                          for(i=1;i<yyleng-1;i++)
                           printf("%c",yytext[i]);
                        }
"/*"[-_a-zA-Z \n]+"*/"
                        {i=0;
                         printf("\n%d\t%d\t\tcomment\t\t",l,t);t++;
                         for(i=2;i<yyleng-2;i++)
                         {
                           if(yytext[i]=='\n')    yytext[i]=' ';
                           printf("%c",yytext[i]);
                         }
                        }
{ffile}                 {printf("\n%d\t%d\t\tspecial char\t\"",l,t);t++;
                          printf("\n%d\t%d\t\tfile\t\t",l,t);t++;
                          for(i=1;i<yyleng-1;i++)
                          printf("%c",yytext[i]);
                          printf("\n%d\t%d\t\tspecial char\t\"",l,t);t++;}
```

```
"printf("[a-zA-Z -_]+")"
                        {printf("\n%d\t%d\t\tfunction\tprintf",l,t);t++;
                         printf("\n%d\t%d\t\tspecial char\t(",l,t);t++;
                         printf("\n%d\t%d\t\toutput\t\t",l,t);t++;
                         for(i=7;i<yyleng-1;i++)
                          printf("%c",yytext[i]);
                         printf("\n%d\t%d\t\tspecial char\t)",l,t);t++;
                        }
("#include<"{hfile}">")|("#include\""[a-z]+"\."{file}"\"")
                        {i=0;
                         printf("\n%d\t%d\t\tpreprocessor\t#",l,t);t++;
                         printf("\n%d\t%d\t\tidentifier\tinclude",l,t);t++;
                         printf("\n%d\t%d\t\tspecial char\t%c",l,t,yytext[8]);t++;
                         printf("\n%d\t%d\t\theader file\t",l,t);t++;
                         for(i=9;i<yyleng-1;i++)
                         printf("%c",yytext[i]);
                         printf("\n%d\t%d\t\tspecial char\t%c",l,t,yytext[i]);t++;
                        }
"#define "[a-z]+" "[a-zA-Z0-9]+
                        {i=0;
                         printf("\n%d\t%d\t\tpreprocessor\t#",l,t);t++;
                         printf("\n%d\t%d\t\tidentifier\tdefine",l,t);t++;
                         printf("\n%d\t%d\t\tidentifier\t",l,t);t++;
                         for(i=8;yytext[i]!=' ';i++)
                          printf("%c",yytext[i]);
                         printf("\n%d\t%d\t\tconstant\t",l,t);t++; i++;
                         for(;i<yyleng;i++)
                          printf("%c",yytext[i]);
                        }
[\n]                    {l++;}

%%

int yywrap()
{       return 1;       }
int main()
{
        yyin=fopen("file.txt","r");
        printf("Line\tToken no\tToken name\tLexeme \n");
        yylex();
        printf("\n");
}
```

Testing:

Input:

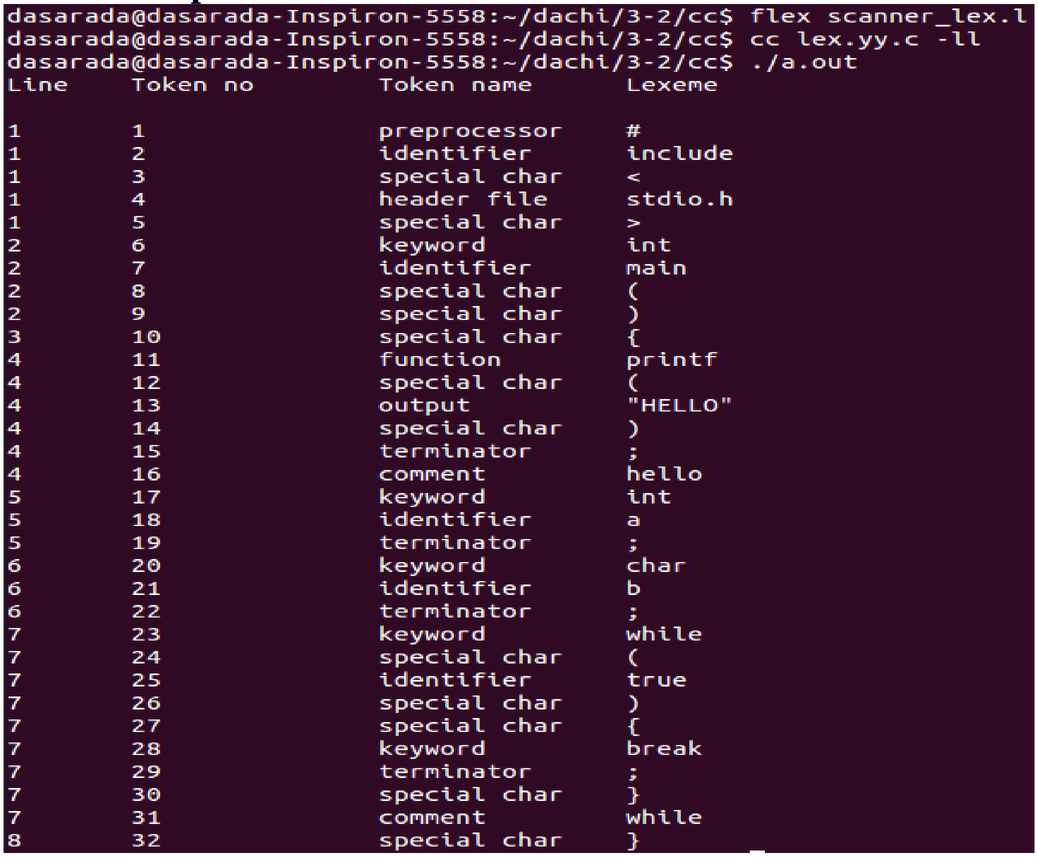
```
//input.c

#include<stdio.h>
int main()
{
        printf("HELLO");//hello
        int a;
        char b;
        while(true){break;}/*while*/
}
```

Expected Output:

```
Line    Token no    Token name      Lexeme
1       1           preprocessor    #
1       2           identifier      include
1       3           special char    <
1       4           header file     stdio.h
1       5           special char    >
2       6           keyword         int
2       7           identifier      main
2       8           special char    (
2       9           special char    )
3       10          special char    {
4       11          function        printf
4       12          special char    (
4       13          output          "HELLO"
4       14          special char    )
4       15          terminator      ;
4       16          comment         hello
5       17          keyword         int
5       18          identifier      a
5       19          terminator      ;
6       20          keyword         char
6       21          identifier      b
6       22          terminator      ;
7       23          keyword         while
7       24          special char    (
```

7	25	identifier	true
7	26	special char	)
7	27	special char	{
7	28	keyword	break
7	29	terminator	;
7	30	special char	}
7	31	comment	while
8	32	special char	}

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ flex scanner_lex.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ cc lex.yy.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Line     Token no          Token name        Lexeme

1        1                 preprocessor      #
1        2                 identifier        include
1        3                 special char      <
1        4                 header file       stdio.h
1        5                 special char      >
2        6                 keyword           int
2        7                 identifier        main
2        8                 special char      (
2        9                 special char      )
3        10                special char      {
4        11                function          printf
4        12                special char      (
4        13                output            "HELLO"
4        14                special char      )
4        15                terminator        ;
4        16                comment           hello
5        17                keyword           int
5        18                identifier        a
5        19                terminator        ;
6        20                keyword           char
6        21                identifier        b
6        22                terminator        ;
7        23                keyword           while
7        24                special char      (
7        25                identifier        true
7        26                special char      )
7        27                special char      {
7        28                keyword           break
7        29                terminator        ;
7        30                special char      }
7        31                comment           while
8        32                special char      }
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy – 160114733092

//Program to identify the Octal or Hexadecimal number using LEX.
%{
#include<stdio.h>
#include<string.h>
%}

%%

[0]                     printf("binary or decimal");
[10]*                   printf("binary");
[1-9][0-9]*             printf("decimal");
[0][0-7]+               printf("octal");
[0][xX][0-9a-fA-F]+     printf("hexadecimal");
[\n] return 0;

%%
int yywrap()
{
        return 1;
}

int main()
{
        printf("Enter a string\n");
        yylex();
}
```

Testing:

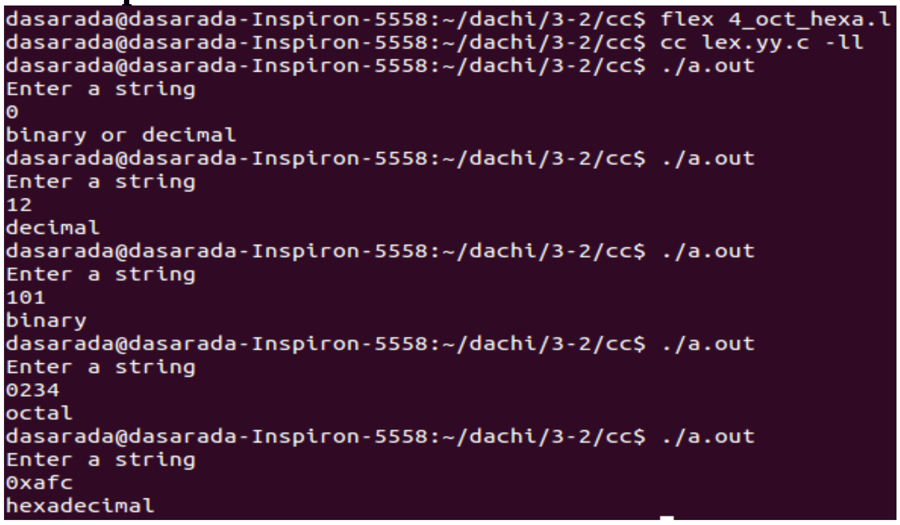
Input:

```
0
12
101
0234
0xafc
```

Expected Output:

```
Enter a string
0
binary or decimal

Enter a string
12
decimal

Enter a string
101
binary

Enter a string
0234
octal

Enter a string
0xafc
hexadecimal
```

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ flex 4_oct_hexa.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ cc lex.yy.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
0
binary or decimal
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
12
decimal
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
101
binary
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
0234
octal
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
0xafc
hexadecimal
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy - 160114733092
// Program to capitalize the input string using LEX.
%{
#include<stdio.h>
%}

%%

[A-Z] {printf("%c",yytext[0]);}
[a-z] {printf("%c",yytext[0]-32);}
[\n]  {return 0;}

%%
int yywrap()
{
	return 1;
}

int main()
{
	printf("Enter a string\n");
	yylex();
}
```

Testing:

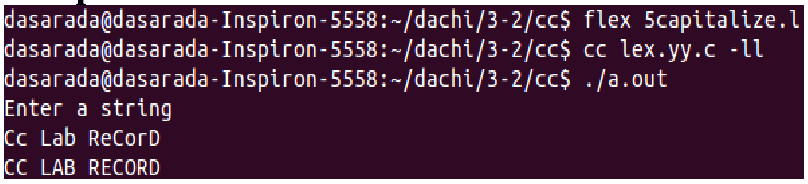
Input:

Cc Lab ReCorD

Expected Output:

```
Enter a string
Cc Lab ReCorD
CC LAB RECORD
```

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ flex 5capitalize.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ cc lex.yy.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
Cc Lab ReCorD
CC LAB RECORD
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy - 160114733092
// Program to find real precision numbers using LEX.
%{
#include<stdio.h>
#include<string.h>
int f,i,j;
%}

%%

[+-]?[0-9]+   {printf("\n%s is an integer!!!",yytext);}
[+-]?[0-9]*[.][0-9]+
        {f=0; for(i=0;i<yyleng;i++)
          if(yytext[i]=='.')
           { j=i+1; break;}
          for(;j<yyleng;j++)
           f++;
          printf("\n%s is a floating number with a precision of %d!!!",yytext,f);}
[0-9a-zA-Z]+[.][0-9+-.a-zA-Z]+          {printf("\ninvalid!!!");}
[\n]              {return 0;}

%%

int main()
{
printf("Enter a number :\n");
yylex();
}
int yywrap()
{
return 1;
}
```

Testing:

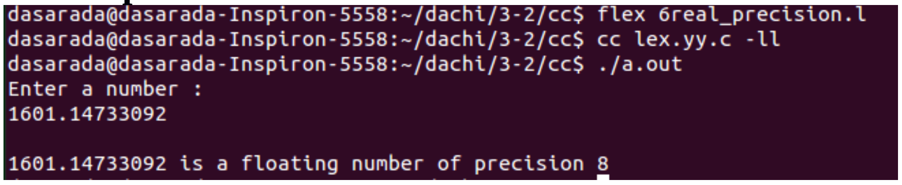
Input:

```
1601.14733092
```

Expected Output:

```
Enter a number :
1601.14733092
1601.14733092 is a floating number of precision 8
```

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ flex 6real_precision.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ cc lex.yy.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a number :
1601.14733092

1601.14733092 is a floating number of precision 8
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy - 160114733092
//Program to count the number of vowels and consonants in a given string using
Lex.
%{
#include<stdio.h>
int vowel=0;
int cons=0;
%}

%%
[aeiouAEIOU] {vowel++;}
[a-zA-Z] {cons++;}
[\n]  { printf("\nVowels=%d and Consonants=%d\n",vowel,cons); return 0;}
%%

int yywrap()
{
	return 1;
}
int main()
{
	printf("Enter a string\n");
	yylex();
}
```

Testing:

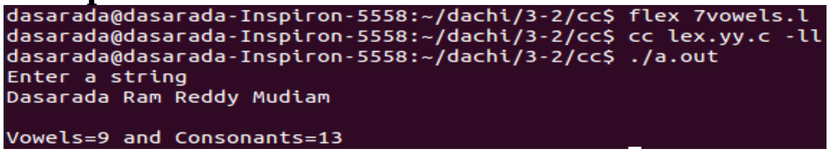
Input:

```
Dasarada Ram Reddy Mudiam
```

Expected Output:

```
Enter a string
Dasarada Ram Reddy Mudiam
Vowels=9 and Consonants=13
```

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ flex 7vowels.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ cc lex.yy.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc$ ./a.out
Enter a string
Dasarada Ram Reddy Mudiam

Vowels=9 and Consonants=13
```

Result:

Successfully executed the program.

Program:

```
//Dasarada Ram Reddy – 160114733092

//Program to implement calculator using Yacc tool.
//calci.l

%{
        #include"y.tab.h"
%}

%%
[0-9]+ {yylval.dval=atoi(yytext);return digit;}
\n|. return yytext[0];
%%

//calci.y

%{
#include<stdio.h>
%}

%union
{
        int dval;
}

%token  <dval> digit
%type <dval> expr
%type <dval> expr1

%%
line:expr '\n' {printf("%d\n",$1);}
    ;
expr:expr '+' expr1 {$$=$1+$3;}
    |expr '-' expr1 {$$=$1-$3;}
    |expr '*' expr1 {$$=$1*$3;}
    |expr '/' expr1 {$$=$1/$3;}
    |expr1
    ;
expr1: '('expr')' {$$=$2;}
     | digit
     ;
%%
```

```
int main()
{
        yyparse();
}
yyerror(char *s)
{
        printf("%s",s);
}
```

Testing:

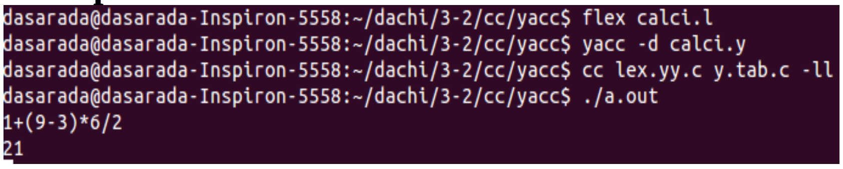
Input:

```
1+(9-3)*6/2
```

Expected Output:

```
1+(9-3)*6/2
21
```

Actual Output:



```
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/yacc$ flex calci.l
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/yacc$ yacc -d calci.y
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/yacc$ cc lex.yy.c y.tab.c -ll
dasarada@dasarada-Inspiron-5558:~/dachi/3-2/cc/yacc$ ./a.out
1+(9-3)*6/2
21
```

Result:

Successfully executed the program.