1. Design a Lexical analyzer for the given language. The lexical analyzer should ignore redundant spaces, tabs and new lines. It should also ignore comments. Although the syntax specification states that identifiers can be arbitrarily long, you may restrict the length to some reasonable value.

```
#include<stdio.h>
int isKeyword(char *str)
{
   char k[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"};
   int i;
   for(i=0;i<32;i++)
   if(strcmp(k[i],str)==0)
   return 1;
return 0;
}
int isFunction(char *str)
{
   if(strcmp(str,"main")==0||strcmp(str,"printf")==0)
   return 1;
return 0;
}
main()
{
int kc,lno=1,sno=0;
char fn[20],c,buf[30];
FILE *fp;
printf("\nEnter the file name:");
scanf("%s",fn);
```

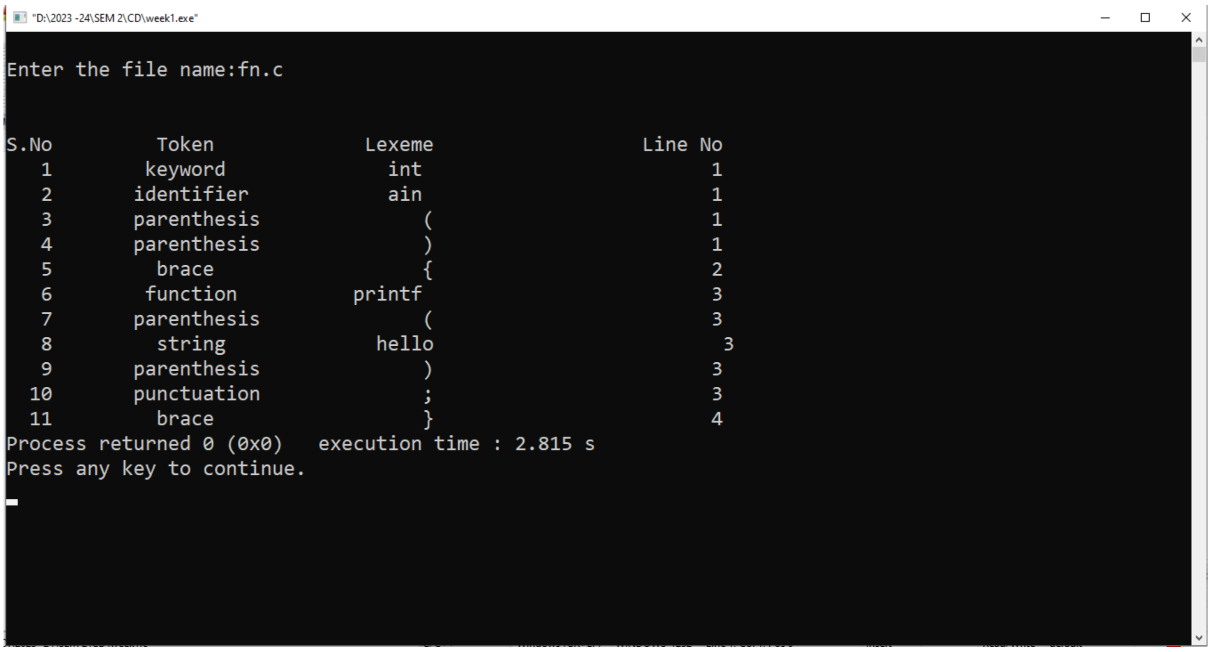
```
printf("\n\nS.No        Token            Lexeme                 Line No");
fp=fopen(fn,"r");
while((c=fgetc(fp))!=EOF)
{
        if(isalpha(c))
        {
                buf[kc=0]=c;
                while(isalnum(c=fgetc(fp)))
                {
                        buf[++kc]=c;
                }
                buf[++kc]='\0';
                if(isKeyword(buf))
                        printf("\n%4d       keyword         %7s      %20d",++sno,buf,lno);
                else if(isFunction(buf))
                        printf("\n%4d       function        %7s      %20d",++sno,buf,lno);
                else
                        printf("\n%4d      identifier       %7s      %20d",++sno,buf,lno);
    }
    else if(isdigit(c))
    {
      buf[kc=0]=c;
      while(isdigit(c=fgetc(fp)))
      buf[++kc]=c;
      buf[++kc]='\0';
      printf("\n%4d        number         %7s      %7d",++sno,buf,lno);
    }
if(c=='('||c==')')
   printf("\n%4d      parenthesis        %6c                %7d",++sno,c,lno);
else if(c=='{'||c=='}')
```

```
   printf("\n%4d        brace            %6c                 %7d",++sno,c,lno);
else if(c=='['||c==']')
   printf("\n%4d      array index        %6c                %7d",++sno,c,lno);
else if(c==','||c==';')
   printf("\n%4d      punctuation        %6c                %7d",++sno,c,lno);
else if(c=='"')
{
          kc=-1;
          while((c=fgetc(fp))!='"')
          buf[++kc]=c;
          buf[++kc]='\0';
          printf("\n%4d        string          %7s      %20d",++sno,buf,lno);
}
else if(c==' ')
   c=fgetc(fp);
else if(c=='\n')
   ++lno;
else
   printf("\n%4d        operator         %6c                %7d",++sno,c,lno);
}
fclose(fp);
}
```

fn.c

```
int main()
{
   printf("hello");
}
```

Output :



"D:\2023 -24\SEM 2\CD\week1.exe"

```
Enter the file name:fn.c


S.No         Token             Lexeme                 Line No
   1        keyword             int                         1
   2       identifier           ain                         1
   3       parenthesis             (                        1
   4       parenthesis             )                        1
   5         brace                 {                        2
   6        function          printf                        3
   7       parenthesis             (                        3
   8         string             hello                        3
   9       parenthesis             )                        3
  10       punctuation             ;                        3
  11         brace                 }                        4
Process returned 0 (0x0)   execution time : 2.815 s
Press any key to continue.
```