

Exp. No. 3

Design a lexical Analyzer for given language should ignore the redundant spaces, tabs and new lines and ignore comments using C

Program:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

int isKeyword(char buffer[]){
char keywords[32][10] =
{"main","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",
"unsigned","void","printf","while"};
int i, flag = 0;
for(i = 0; i < 32; ++i)
{
if(strcmp(keywords[i], buffer) == 0)
{
flag = 1;
break;
}
}
return flag;
}

int main()
{
char ch, buffer[15], operators[] = "+-*/%=";
FILE *fp;
int i,j=0;
fp = fopen("flex_input.txt","r");
```

```

if(fp == NULL){
printf("error while opening the file\n");
exit(0);
}
while((ch = fgetc(fp)) != EOF){
    for(i = 0; i < 6; ++i){
        if(ch == operators[i])
            printf("%c is operator\n", ch);
    }

    if(isalnum(ch)){
        buffer[j++] = ch;
    }
    else if((ch == ' ' || ch == '\n') && (j != 0)){
        buffer[j] = '\0';
        j = 0;

        if(isKeyword(buffer) == 1)
            printf("%s is keyword\n", buffer);
        else
            printf("%s is identifier\n", buffer);
    }
}
fclose(fp);
return 0;
}

```

Input: flex_input.txt

```

main ( )
{
    int a, b, c ;
    c = b + c;
    printf ( "%d" ,c ) ;
}

```

OUTPUT:

```
C:\Users\hp\Documents\Com x + -
main is keyword
int is keyword
a is identifier
b is identifier
c is identifier
c is identifier
= is operator
b is identifier
+ is operator
c is identifier
printf is keyword
% is operator
d is identifier
c is identifier

-----
Process exited after 0.01637 seconds with return value 0
Press any key to continue . . .
```