Cycle 2

Malavika R Vikraman

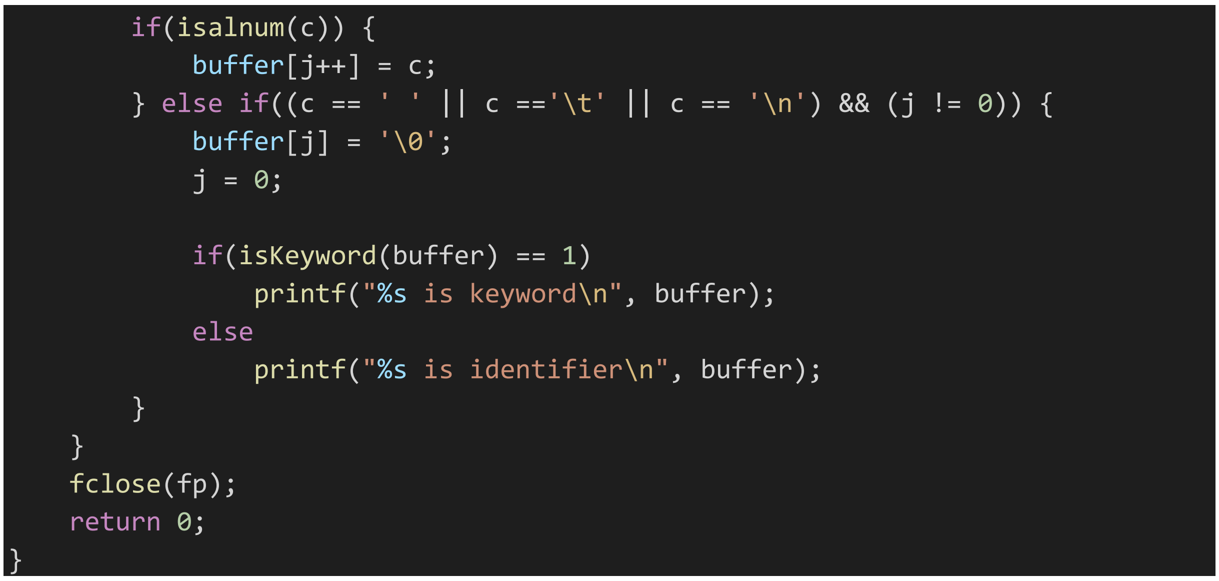
CS7B – Roll No :31

Question 1

Design and implement a lexical analyzer for given language using C and the lexical analyzer should ignore redundant spaces, tabs and newlines.

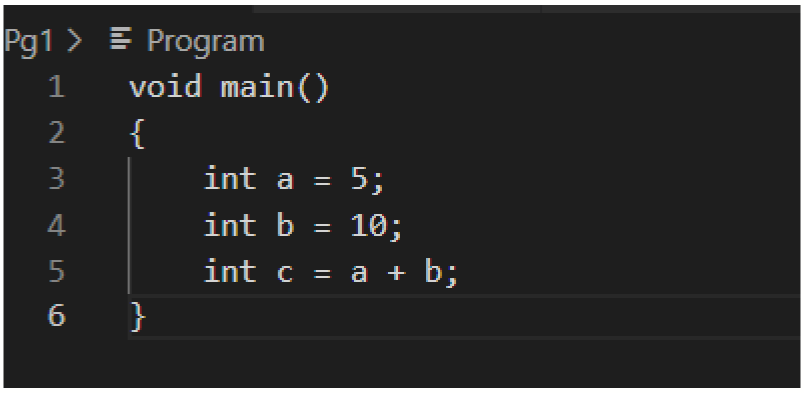
Program :

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

int isKeyword(char buffer[]){
    char keywords[32][10] = 
{"auto","break","case","char","const","continue","default",
                            "do","double","else","enum","extern","float","for"
,"goto",
                            "if","int","long","register","return","short","sig
ned",
                            "sizeof","static","struct","switch","typedef","uni
on",
                            "unsigned","void","volatile","while"};
    int i;
    for(i = 0; i < 32; ++i){
        if(strcmp(keywords[i], buffer) == 0){
            return 1;
        }
    }
    return 0;
}

int main() {
    char c, buffer[31], operators[] = "+-*/%=";
    FILE *fp;
    int i, j=0;

    fp = fopen("Program","r");

    if(fp == NULL){
        printf("Error while opening the file\n");
        exit(0);
    }

    while((c = fgetc(fp)) != EOF) {
        for(i = 0; i < 6; ++i){
            if(c == operators[i])
                printf("%c is operator\n", c);
        }
```



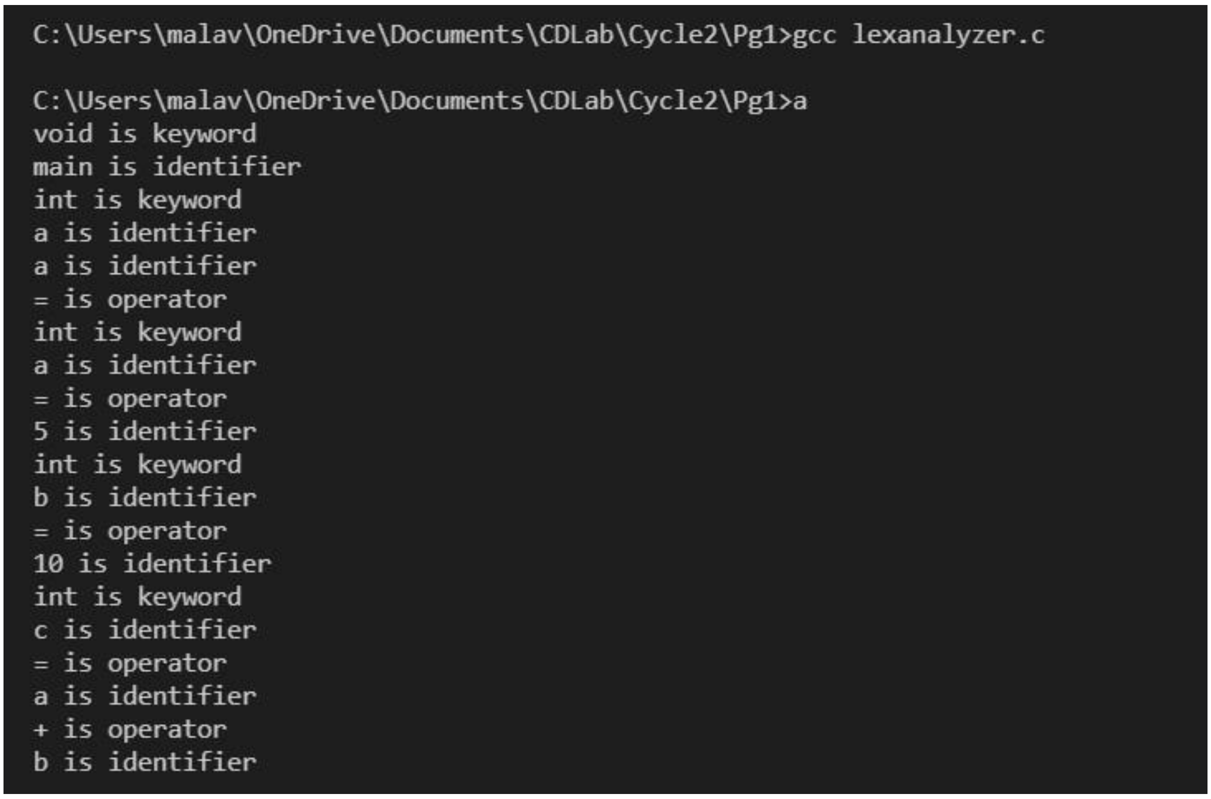
```c
        if(isalnum(c)) {
            buffer[j++] = c;
        } else if((c == ' ' || c =='\t' || c == '\n') && (j != 0)) {
            buffer[j] = '\0';
            j = 0;

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

Input :



```c
void main()
{
    int a = 5;
    int b = 10;
    int c = a + b;
}
```

Output :



```
C:\Users\malav\OneDrive\Documents\CDLab\Cycle2\Pg1>gcc lexanalyzer.c

C:\Users\malav\OneDrive\Documents\CDLab\Cycle2\Pg1>a
void is keyword
main is identifier
int is keyword
a is identifier
a is identifier
= is operator
int is keyword
a is identifier
= is operator
5 is identifier
int is keyword
b is identifier
= is operator
10 is identifier
int is keyword
c is identifier
= is operator
a is identifier
+ is operator
b is identifier
```