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:

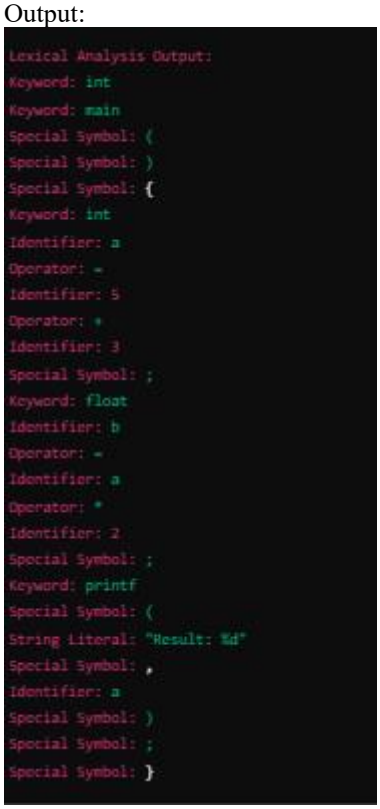
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

// Function to check if a string is a keyword
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"
   };
   for (int i = 0; i < 32; ++i) {
      if (strcmp(keywords[i], buffer) == 0) {
         return 1; // It is a keyword
      }
   }
   return 0;
}

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

   // Open the file
   fp = fopen("flex_input.txt", "r");
   if (fp == NULL) {
      printf("Error while opening the file\n");
      exit(1);
   }

   printf("Lexical Analysis Output:\n");

   while ((ch = fgetc(fp)) != EOF) {
      // Check for operators
      for (i = 0; i < strlen(operators); i++) {
         if (ch == operators[i]) {
            printf("Operator: %c\n", ch);
         }
      }

      // Check for alphanumeric characters (identifiers or keywords)
      if (isalnum(ch)) {
         buffer[j++] = ch;
      } else if ((ch == ' ' || ch == '\n' || ch == ';' || ch == '(' || ch == ')') && (j != 0)) {
         buffer[j] = '\0'; // Null terminate the string
         j = 0;

         // Check if it's a keyword
         if (isKeyword(buffer)) {
            printf("Keyword: %s\n", buffer);
```

```
            } else {
                printf("Identifier: %s\n", buffer);
            }
        }

        // Check for special symbols
        if (ch == ';' || ch == '(' || ch == ')') {
            printf("Special Symbol: %c\n", ch);
        }
    }

    fclose(fp); // Close the file
    return 0;
}
```

Input:

```
int main() {
    int a = 5 + 3;
    float b = a * 2;
    printf("Result: %d", a);
}
```

Output:



```
Lexical Analysis Output:
Keyword: int
Keyword: main
Special Symbol: (
Special Symbol: )
Special Symbol: {
Keyword: int
Identifier: a
Operator: =
Identifier: 5
Operator: +
Identifier: 3
Special Symbol: ;
Keyword: float
Identifier: b
Operator: =
Identifier: a
Operator: *
Identifier: 2
Special Symbol: ;
Keyword: printf
Special Symbol: (
String Literal: "Result: %d"
Special Symbol: ,
Identifier: a
Special Symbol: )
Special Symbol: ;
Special Symbol: }
```