## Develop a lexical Analyzer to identify identifiers, constants, operators using C program.

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
// Function to check if a character is an operator
int isOperator(char ch) {
  char operators[] = "+-*/%=<>!&|";
  for (int i = 0; i < strlen(operators); i++) {
    if (ch == operators[i]) return 1;
  }
  return 0;
}
// Function to check if a given string is a keyword
int isKeyword(char *word) {
  char *keywords[] = {"int", "float", "char", "if", "else", "while", "for", "return", "void", "do",
"switch", "case"};
  int numKeywords = sizeof(keywords) / sizeof(keywords[0]);
  for (int i = 0; i < numKeywords; i++) {
    if (strcmp(word, keywords[i]) == 0) return 1;
  }
  return 0;
}
// Function to check if a string is a number
int isNumber(char *word) {
  for (int i = 0; i < strlen(word); i++) {
    if (!isdigit(word[i]) && word[i] != '.') return 0;
  }
```

```
return 1;
}
// Function to analyze a given input string
void lexicalAnalyzer(char *input) {
  int len = strlen(input);
  int i = 0;
  printf("Tokens Identified:\n");
  while (i < len) {
    // Skip whitespace
    if (isspace(input[i])) {
       i++;
       continue;
    }
    // Identifiers & Keywords
    if (isalpha(input[i])) {
       char word[50];
       int j = 0;
       while (isalnum(input[i])) {
         word[j++] = input[i++];
       }
       word[j] = '\0';
       if (isKeyword(word)) {
         printf("Keyword: %s\n", word);
       } else {
         printf("Identifier: %s\n", word);
       }
```

```
}
 // Numbers (Constants)
  else if (isdigit(input[i])) {
            char num[50];
            int j = 0;
            while (isdigit(input[i]) || input[i] == '.') {
                       num[j++] = input[i++];
            }
            num[j] = '\0';
            printf("Constant: %s\n", num);
}
 // Operators
  else if (isOperator(input[i])) {
            char op[3] = \{input[i], '\0', '\0'\};
            // Handle multi-character operators (==, !=, <=, >=, &&, ||)
            if ((input[i] == '=' \mid | \ input[i] == '!' \mid | \ input[i] == '<' \mid | \ input[i] == '>') \&\& \ input[i + 1] == '=') \{ (input[i] == '>') \&\& \ input[i] == '=' \mid | \ input[i] == '
                       op[1] = '=';
                       i++;
            } else if ((input[i] == '&' || input[i] == '|') && input[i + 1] == input[i]) {
                       op[1] = input[i];
                      i++;
            }
            printf("Operator: %s\n", op);
            i++;
}
 // Special characters (skip them)
```

```
else {
             printf("Symbol: %c\n", input[i]);
             i++;
         }
    }
}
// Main function to take input and call the lexical analyzer
int main() {
     char input[100];
     printf("Enter an expression: ");
    fgets(input, sizeof(input), stdin);
    lexicalAnalyzer(input);
     return 0;
}
Input:
int x = 10 + 20;
Output:
   PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
  PS C:\Users\valli> & 'c:\Users\valli\.vscode\extensions\ms-vscode.cpptools-1.22.11-win32-x64\debugAdapters\bin\WindowsDebugLauncher.exe' '--stdin=Mi crosoft-MIEngine-In-wy0j4thu.it2' '--stdout=Microsoft-MIEngine-Out-zwoa1iwj.v1p' '--stderr=Microsoft-MIEngine-Error-ehntwynz.043' '--pid=Microsoft-MIEngine-Pid-wgsufh0g.oxv' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi' Enter an expression: int x = 10 + 20;
Tokens Identified:
Keyword: int
Identifier: x
Operator: =
   Operator: = Constant: 10
   Operator: +
Constant: 20
Symbol: ;
```