**CSA1443-COMPILER DESIGN FOR INTRAPROCEDURAL ANALYSIS**

**NAME: Ajay Kumar J**

**REG NO: 192372052**

**PROGRAM 6  
Develop a lexical Analyzer to test whether a given identifier is valid or not using C.**

**Aim:**

To develop a lexical analyzer in C that checks whether a given identifier is valid according to the rules of the C programming language.

**Code:**

#include <stdio.h>

#include <ctype.h>

#include <string.h>

// List of C keywords

const char \*keywords[] = {

"auto", "break", "case", "char", "const", "continue", "default", "do", "double",

"else", "enum", "extern", "float", "for", "goto", "if", "inline", "int", "long",

"register", "restrict", "return", "short", "signed", "sizeof", "static", "struct",

"switch", "typedef", "union", "unsigned", "void", "volatile", "while", "\_Alignas",

"\_Alignof", "\_Atomic", "\_Bool", "\_Complex", "\_Generic", "\_Imaginary", "\_Noreturn",

"\_Static\_assert", "\_Thread\_local"

};

// Function to check if a given string is a keyword

int isKeyword(char \*str) {

int n = sizeof(keywords) / sizeof(keywords[0]);

for (int i = 0; i < n; i++) {

if (strcmp(str, keywords[i]) == 0)

return 1; // It is a keyword

}

return 0;

}

// Function to check if a given string is a valid identifier

int isValidIdentifier(char \*str) {

// Check if it's a keyword

if (isKeyword(str))

return 0;

// Check if the first character is a letter or underscore

if (!isalpha(str[0]) && str[0] != '\_')

return 0;

// Check remaining characters

for (int i = 1; str[i] != '\0'; i++) {

if (!isalnum(str[i]) && str[i] != '\_')

return 0;

}

return 1;

}

// Main function

int main() {

char identifier[50];

printf("Enter an identifier: ");

scanf("%s", identifier);

if (isValidIdentifier(identifier))

printf("\"%s\" is a valid identifier.\n", identifier);

else

printf("\"%s\" is not a valid identifier.\n", identifier);

return 0;

}

**Output:**

****