**Exp 3:**

**Token Separation using C**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define MAX\_LEN 100

// List of C keywords

const char \*keywords[] = {

"int", "float", "double", "char", "return", "if", "else", "while", "for", "void", "break", "continue"

};

// Function to check if a string is a keyword

int isKeyword(char \*str) {

for (int i = 0; i < sizeof(keywords) / sizeof(keywords[0]); i++) {

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

return 1; // It's a keyword

}

}

return 0; // Not a keyword

}

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

int isIdentifier(char \*str) {

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

return 0; // Identifiers must start with a letter or underscore

}

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

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

return 0; // Identifiers can only contain letters, digits, and underscores

}

}

return 1;

}

// Function to classify and print tokens

void classifyToken(char \*token) {

if (isKeyword(token)) {

printf("Keyword: %s\n", token);

} else if (isIdentifier(token)) {

printf("Identifier: %s\n", token);

} else if (strchr("+-\*/%=<>!&|^~", token[0]) != NULL) {

printf("Operator: %s\n", token);

} else if (strchr("()[]{};,.#", token[0]) != NULL) {

printf("Punctuation: %s\n", token);

} else {

printf("Unknown Token: %s\n", token);

}

}

// Function to tokenize and classify the input

void tokenize(char \*str) {

char \*token = strtok(str, " \t\n\r"); // Split the string by spaces, tabs, or newlines

while (token != NULL) {

classifyToken(token);

token = strtok(NULL, " \t\n\r"); // Get the next token

}

}

int main() {

char input[MAX\_LEN];

printf("Enter a C code snippet: ");

fgets(input, MAX\_LEN, stdin); // Read input from the user

tokenize(input); // Process and classify tokens

return 0;

}

