PROGRAM 3

1. Design a lexical Analyzer for given language should ignore the redundant spaces, tabs and new lines and ignore comments using C

CODE:

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define MAX\_TOKEN\_LENGTH 100

char keywords[][10] = {"int", "float", "if", "else", "while", "return"};

int isKeyword(char \*str) {

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

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

return 1;

}

}

return 0;

}

void removeRedundantSpaces(char \*input, char \*output) {

int i = 0, j = 0;

while (input[i] != '\0') {

if (isspace(input[i])) {

if (j == 0 || isspace(output[j - 1]) == 0) {

output[j++] = ' ';

}

} else {

output[j++] = input[i];

}

i++;

}

output[j] = '\0';

}

void processCode(char \*code) {

char token[MAX\_TOKEN\_LENGTH];

int i = 0, j = 0;

printf("Tokens:\n");

while (code[i] != '\0') {

if (isspace(code[i])) {

if (j > 0) {

token[j] = '\0';

if (isKeyword(token)) {

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

} else if (isalpha(token[0]) || token[0] == '\_') {

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

} else {

printf("Literal/Other: %s\n", token);

}

j = 0;

}

} else if (code[i] == '/' && code[i + 1] == '/') {

while (code[i] != '\0' && code[i] != '\n') i++;

} else if (code[i] == '/' && code[i + 1] == '\*') {

i += 2;

while (code[i] != '\0' && !(code[i] == '\*' && code[i + 1] == '/')) i++;

i++;

} else {

token[j++] = code[i];

}

i++;

}

if (j > 0) {

token[j] = '\0';

if (isKeyword(token)) {

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

} else if (isalpha(token[0]) || token[0] == '\_') {

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

} else {

printf("Literal/Other: %s\n", token);

}

}

}

int main() {

char input[] = "int main() { int a = 10; // Variable declaration\n"

"/\* Multiline Comment \*/\n"

"float b = 20.5;\n"

"}";

char processedCode[1000];

removeRedundantSpaces(input, processedCode);

processCode(processedCode);

return 0;

}

OUTPUT:

Tokens:

Keyword: int

Identifier: main()

Literal/Other: {

Keyword: int

Identifier: a

Literal/Other: =

Literal/Other: 10;

Literal/Other: ��