## OBJECTIVE-2

**AIM:-Write a program in 'C' to recognize the following tokens and display the message with the token name :**
**i. Identifiers: A string starting with an underscore or a letter and followed by any number of underscores, letters and digits . Identifiers with two leading underscores(__) are disallowed.**
**ii. Keywords: short ,sizeof,int, float, double, bool, char, signed, unsigned, for, while, do, return, struct,const, void, switch, break, case, continue, goto, long ,static, union,default**
**iii. Signed and unsigned Integer constants: 12, 0, 3456, +56, -234 etc.**
**iv. Signed and unsigned Floating-point constant: 1.2, 4.25, -0.35 etc.**
**v. Arithmetic operators: +, -, *, /, %, ++, --**
**vi. Assignment operators: =, +=, -=, *=, /=**
**vii. Relational operators: <,>, <=, >=, ==**
**viii. Special symbols: ; ( ) ,(comma) [ ] { }**

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int isDelimiter(char ch)
{
        if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
                ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
                ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
                ch == '[' || ch == ']' || ch == '{' || ch == '}')
                return (1);
        return (0);
}

int isOperator(char ch)
{
   if (ch == '+' || ch == '-' || ch == '*' ||
      ch == '/' || ch == '>' || ch == '<' ||
      ch == '=')
      return (1);
   return (0);
}
```

```
int validIdentifier(char* str)
{
	if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
		str[0] == '3' || str[0] == '4' || str[0] == '5' ||
		str[0] == '6' || str[0] == '7' || str[0] == '8' ||
		str[0] == '9' || str[0]=='_' &&str[1]=='_'||isDelimiter(str[0]) == 1)
		return (0);
	return (1);
}

int isKeyword(char* str)
{
	if (!strcmp(str, "if") || !strcmp(str, "else")
		|| !strcmp(str, "while") || !strcmp(str, "do")
		|| !strcmp(str, "break") || !strcmp(str, "for")
		|| !strcmp(str, "union") ||!strcmp(str, "default")
		|| !strcmp(str, "continue") || !strcmp(str, "int")
		|| !strcmp(str, "double") || !strcmp(str, "float")
		|| !strcmp(str, "return") || !strcmp(str, "char")
		|| !strcmp(str, "case")   || !strcmp(str, "bool")
		|| !strcmp(str, "sizeof") || !strcmp(str, "long")
		|| !strcmp(str, "short")  || !strcmp(str, "const")
		|| !strcmp(str, "switch") || !strcmp(str, "unsigned")
		|| !strcmp(str, "signed") || !strcmp(str, "void") || !strcmp(str, "static")
		|| !strcmp(str, "struct") || !strcmp(str, "goto"))
		return (1);
	return (0);
}

int isInteger(char* str)
{
	int i, len = strlen(str);

	if (len == 0)
		return (0);
	for (i = 0; i < len; i++) {
		if (str[i] != '0' && str[i] != '1' && str[i] != '2'
			&& str[i] != '3' && str[i] != '4' && str[i] != '5'
			&& str[i] != '6' && str[i] != '7' && str[i] != '8'
			&& str[i] != '9' || (str[i] == '-' && i > 0))
			return (0);
		}
	return (1);
```

```
}
int isSignedInteger(char* str)
{
	int i, len = strlen(str),j=0;

	if (len == 0)
		return (0);
	for (i = 0; i < len; i++) {
		if (str[i] != '0' && str[i] != '1' && str[i] != '2'
			&& str[i] != '3' && str[i] != '4' && str[i] != '5'
			&& str[i] != '6' && str[i] != '7' && str[i] != '8'
			&& str[i] != '9' && (str[i]!='.'||str[i]!='+'|| str[i]!='-')) {
			return 0;
			}
		}
	return (1);
}

int isRealNumber(char* str)
{
	int i, len = strlen(str);
	int hasDecimal = 0;

	if (len == 0)
		return (0);
	for (i = 0; i < len; i++) {
		if (str[i] != '0' && str[i] != '1' && str[i] != '2'
			&& str[i] != '3' && str[i] != '4' && str[i] != '5'
			&& str[i] != '6' && str[i] != '7' && str[i] != '8'
			&& str[i] != '9' && str[i]!='-'&&str[i]!='+'){
				hasDecimal=0;
			}
		else
		  hasDecimal=1;
	}
	return (hasDecimal);
}

char* subString(char* str, int left, int right)
{
	int i;
	char* subStr = (char*)malloc(sizeof(char) * (right - left + 2));

	for (i = left; i <= right; i++)
```

```
                subStr[i - left] = str[i];
        subStr[right - left + 1] = '\0';
        return (subStr);
}

void parse(char* str)
{
        int left = 0, right = 0;
        int len = strlen(str);

        while (right <= len && left <= right) {
                if (isDelimiter(str[right]) == 0)
                        right++;

                if (isDelimiter(str[right]) == 1 && left == right) {
                        if (isOperator(str[right]) == 1)
                                printf("'%c' IS AN OPERATOR\n", str[right]);

                        right++;
                        left = right;
                } else if (isDelimiter(str[right]) == 1 && left != right
                                || (right == len && left != right)) {
                        char* subStr = subString(str, left, right - 1);

                        if (isKeyword(subStr) == 1)
                                printf("'%s' IS A KEYWORD\n", subStr);

                        else if (isInteger(subStr) == 1)
                                printf("'%s' IS AN UNSIGNED INTEGER\n", subStr);

        else if (isSignedInteger(subStr) == 1)
                                printf("'%s' IS  AN SIGNED INTEGER\n", subStr);

                        else if (isRealNumber(subStr) == 1)
                                printf("'%s' IS A SIGNED FLOAT CONSTANT\n", subStr);

                   else if (isRealNumber(subStr) == 0)
                                printf("'%s' IS AN UNSIGNED FLOAT CONSTANT \n", subStr);

                        else if (validIdentifier(subStr) == 1
                                        && isDelimiter(str[right - 1]) == 0)
                                printf("'%s' IS A VALID IDENTIFIER\n", subStr);

                        else if (validIdentifier(subStr) == 0
```

```
                                        && isDelimiter(str[right - 1]) == 0)
                                    printf("'%s' IS NOT A VALID IDENTIFIER\n", subStr);
                            left = right;
                        }
                }
                return;
}

int main()
{
        char str[100] = "(-0.3) int 3 ";
        parse(str);
        return (0);
}
```

## OUTPUT-

```
'-' IS AN OPERATOR
'0.3' IS A SIGNED FLOAT CONSTANT
'int' IS A KEYWORD
'3' IS AN UNSIGNED INTEGER

--------------------------------
Process exited after 0.06883 seconds with return value 0
Press any key to continue . . .
```