NAME:- Sourav Paul
GROUP:- D2
REG. NO:- 20214056
BRANCH:- CSE DEPT.

ASSIGNMENT - 01

1). Write a program in C which takes a C program as input and display all the keywords in the program?

Ans:-

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<stdbool.h>

int check1(char str[])
{
	if (!strcmp(str, "auto") || !strcmp(str, "default") || !
strcmp(str, "signed") || !strcmp(str, "enum")
		||!strcmp(str, "extern") || !strcmp(str, "for") || !
strcmp(str, "register") || !strcmp(str, "if")
		|| !strcmp(str, "else") || !strcmp(str, "int") || !strcmp(str,
"while") || !strcmp(str, "do")
		|| !strcmp(str, "break") || !strcmp(str, "continue") || !
strcmp(str, "double") || !strcmp(str, "float")
		|| !strcmp(str, "return") || !strcmp(str, "char") || !
strcmp(str, "case") || !strcmp(str, "const")
```

```
        || !strcmp(str, "sizeof") || !strcmp(str, "long") || !
strcmp(str, "short") || !strcmp(str, "typedef")
        || !strcmp(str, "switch") || !strcmp(str, "unsigned") || !
strcmp(str, "void") || !strcmp(str, "static")
        || !strcmp(str, "struct") || !strcmp(str, "goto") || !
strcmp(str, "union") || !strcmp(str, "volatile"))
    {
        return 1;
    }
    else
    {
        return 0;
    }
}

int main()
{
    char str[30];
    int n;
    printf("Enter no. of strings to be checked: ");
    scanf("%d", &n);
    printf("Enter a string to be checked: ");
    int i;
    for(i=0; i<n; i++)
    {
        scanf("%s", &str);
        if(check1(str))
        {
            printf("%s is a keyword.\n", str);
        }
        else
        printf("%s is not a keyword.\n", str);
  }
    return 0;
}
```

/*OUTPUT
Enter no. of strings to be checked: 8
Enter a string to be checked: int a = float b + c
int is a keyword.
a is not a keyword.
= is not a keyword.
float is a keyword.
b is not a keyword.
+ is not a keyword.
c is not a keyword.

Q2). Write a program in C which takes a C program as input and display all the tokens in the program?

Ans:-

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Returns 'true' if the character is a DELIMITER.
bool isDelimiter(char ch)
{
	if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
		ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
		ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
		ch == '[' || ch == ']' || ch == '{' || ch == '}')
		return (true);
	return (false);
}

// Returns 'true' if the character is an OPERATOR.
bool isOperator(char ch)
```

```
{
	if (ch == '+' || ch == '-' || ch == '*' ||
		ch == '/' || ch == '>' || ch == '<' ||
		ch == '=')
		return (true);
	return (false);
}

// Returns 'true' if the string is a VALID IDENTIFIER.
bool 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' || isDelimiter(str[0]) == true)
		return (false);
	return (true);
}

// Returns 'true' if the string is a KEYWORD.
bool isKeyword(char* str)
{
	if (!strcmp(str, "if") || !strcmp(str, "else") ||
		!strcmp(str, "while") || !strcmp(str, "do") ||
		!strcmp(str, "break") ||
		!strcmp(str, "continue") || !strcmp(str, "int")
		|| !strcmp(str, "double") || !strcmp(str, "float")
		|| !strcmp(str, "return") || !strcmp(str, "char")
		|| !strcmp(str, "case") || !strcmp(str, "char")
		|| !strcmp(str, "sizeof") || !strcmp(str, "long")
		|| !strcmp(str, "short") || !strcmp(str, "typedef")
		|| !strcmp(str, "switch") || !strcmp(str, "unsigned")
		|| !strcmp(str, "void") || !strcmp(str, "static")
		|| !strcmp(str, "struct") || !strcmp(str, "goto"))
		return (true);
	return (false);
```

```
}

// Returns 'true' if the string is an INTEGER.
bool isInteger(char* str)
{
	int i, len = strlen(str);

	if (len == 0)
		return (false);
	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 (false);
	}
	return (true);
}

// Returns 'true' if the string is a REAL NUMBER.
bool isRealNumber(char* str)
{
	int i, len = strlen(str);
	bool hasDecimal = false;

	if (len == 0)
		return (false);
	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] == '-' && i > 0))
			return (false);
```

```
            if (str[i] == '.')
                hasDecimal = true;
        }
        return (hasDecimal);
}

// Extracts the SUBSTRING.
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);
}

// Parsing the input STRING.
void parse(char* str)
{
        int left = 0, right = 0;
        int len = strlen(str);

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

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

                        right++;
                        left = right;
```

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

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

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

                else if (isRealNumber(subStr) == true)
                    printf("'%s' IS A REAL NUMBER\n",
subStr);

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

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

int main()
{
        // maximum length of string is 100 here
        char str[100] = "float x = y + 5z; ";

        parse(str); // calling the parse function
```

```
        return (0);
}

/* OUTPUT

'float' IS A KEYWORD
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'y' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'5z' IS NOT A VALID IDENTIFIER
*/
```