

```
// Name: AHAN BANDYOPADHYAY
// Roll No.: 211210008
// CSB353 Compiler Design Lab2
// https://boxofnotes.com/lexical-analyzer-in-c-program-to-detect-tokens/
```

/* Q. Write a C program for generating a lexical analyser to identify the following issues:

1) Keywords:

Examples- for, while, if, printf etc.

2) Identifier:

Examples- variable name, function name etc.

3) Operators:

Examples- '+', '++', '-', etc.

4) Separators:

Examples- ',', ';', etc

```
*/
```

```
//CODE:
```

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

```
// function to identify the delimiters
```

```
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);
}
```

```
// function to identify the separators
```

```
bool isSeparator(char ch){
    if (ch == ' ' || ch == ',' || ch == ';' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}')
        return (true);

    return (false);
}
```

```
// function to identify the operators
```

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

```
// function to identify the keywords in C
```

```
bool isKeyword(char* str)  
{  
  
    if(!strcmp(str, "if") || !strcmp(str, "else") ||  
        !strcmp(str, "while") || !strcmp(str, "do") || !strcmp(str, "for") || !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);  
}
```

```
// function to check if the character is an INTEGER.
```

```
bool isInteger(char* str)  
{  
    int len = strlen(str);  
  
    if (len == 0)  
        return false;  
  
    for (int 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;
}
```

// function to check if the character is an INTEGER.

bool isRealNumber(char* str)

```
{
    int len = strlen(str);

    bool hasDecimal = false;

    if (len == 0)
        return false;

    for (int 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;
}
```

// function to check the validity of the token

bool isValid(char* s){

```
    if (s[0] == '0' || s[0] == '1' || s[0] == '2' || s[0] == '3' || s[0] == '4' || s[0] == '5' || s[0] == '6' || s[0] == '7' || s[0] == '8' ||
        s[0] == '9' || isDelimiter(s[0]))
        return (false);

    return (true);
}
```

```

// 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);
}

```

```

void lexAnalyser(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]);

            if (isSeparator(str[right]) == true)
                printf("'%c' is a SEPARATOR\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 (isValid(subStr) == true
        && isDelimiter(str[right - 1]) == false)
        printf("'%s' is an VALID IDENTIFIER\n", subStr);

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

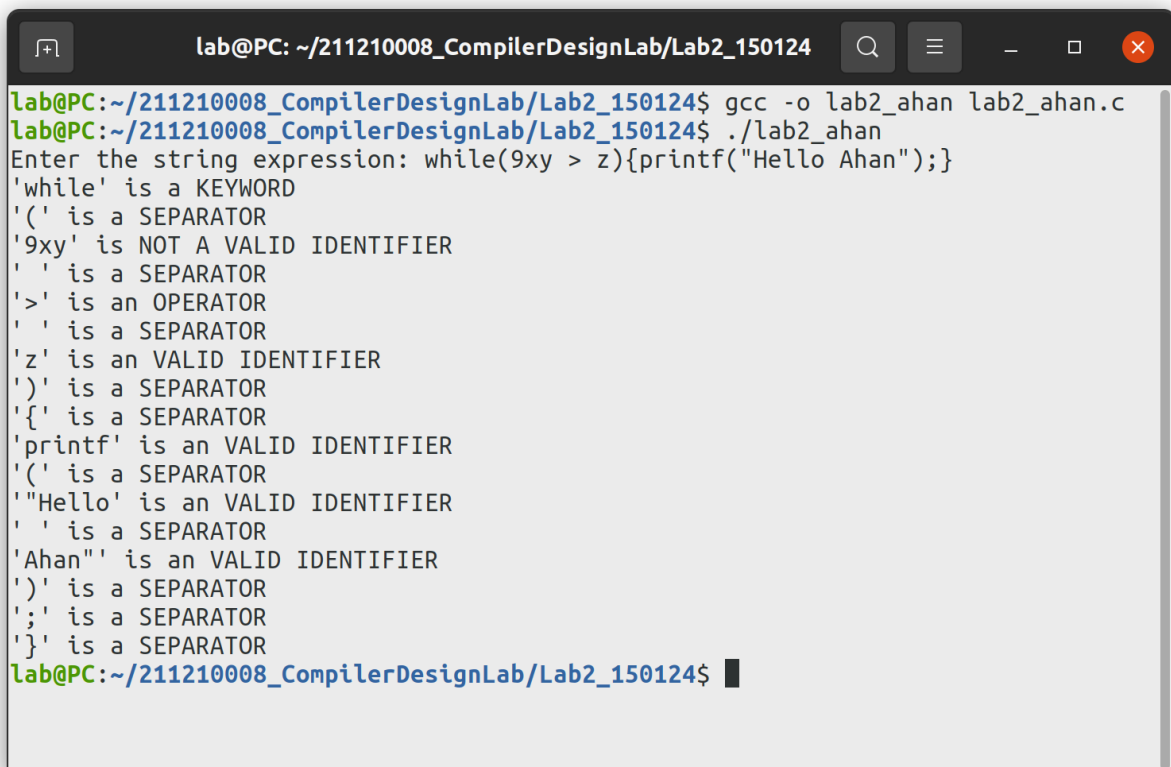
int main(){
    //string of len max 50
    char input[50];

    printf("Enter the string expression: ");
    scanf("%[^\\n]s", input);

    lexAnalyser(input);

    return 0;
}

```



```

lab@PC: ~/211210008_CompilerDesignLab/Lab2_150124
lab@PC:~/211210008_CompilerDesignLab/Lab2_150124$ gcc -o lab2_ahan lab2_ahan.c
lab@PC:~/211210008_CompilerDesignLab/Lab2_150124$ ./lab2_ahan
Enter the string expression: while(9xy > z){printf("Hello Ahan");}
'while' is a KEYWORD
'(' is a SEPARATOR
'9xy' is NOT A VALID IDENTIFIER
' ' is a SEPARATOR
'>' is an OPERATOR
' ' is a SEPARATOR
'z' is an VALID IDENTIFIER
')' is a SEPARATOR
'{' is a SEPARATOR
'printf' is an VALID IDENTIFIER
'(' is a SEPARATOR
'Hello' is an VALID IDENTIFIER
' ' is a SEPARATOR
'Ahan' is an VALID IDENTIFIER
')' is a SEPARATOR
';' is a SEPARATOR
'}' is a SEPARATOR
lab@PC:~/211210008_CompilerDesignLab/Lab2_150124$

```