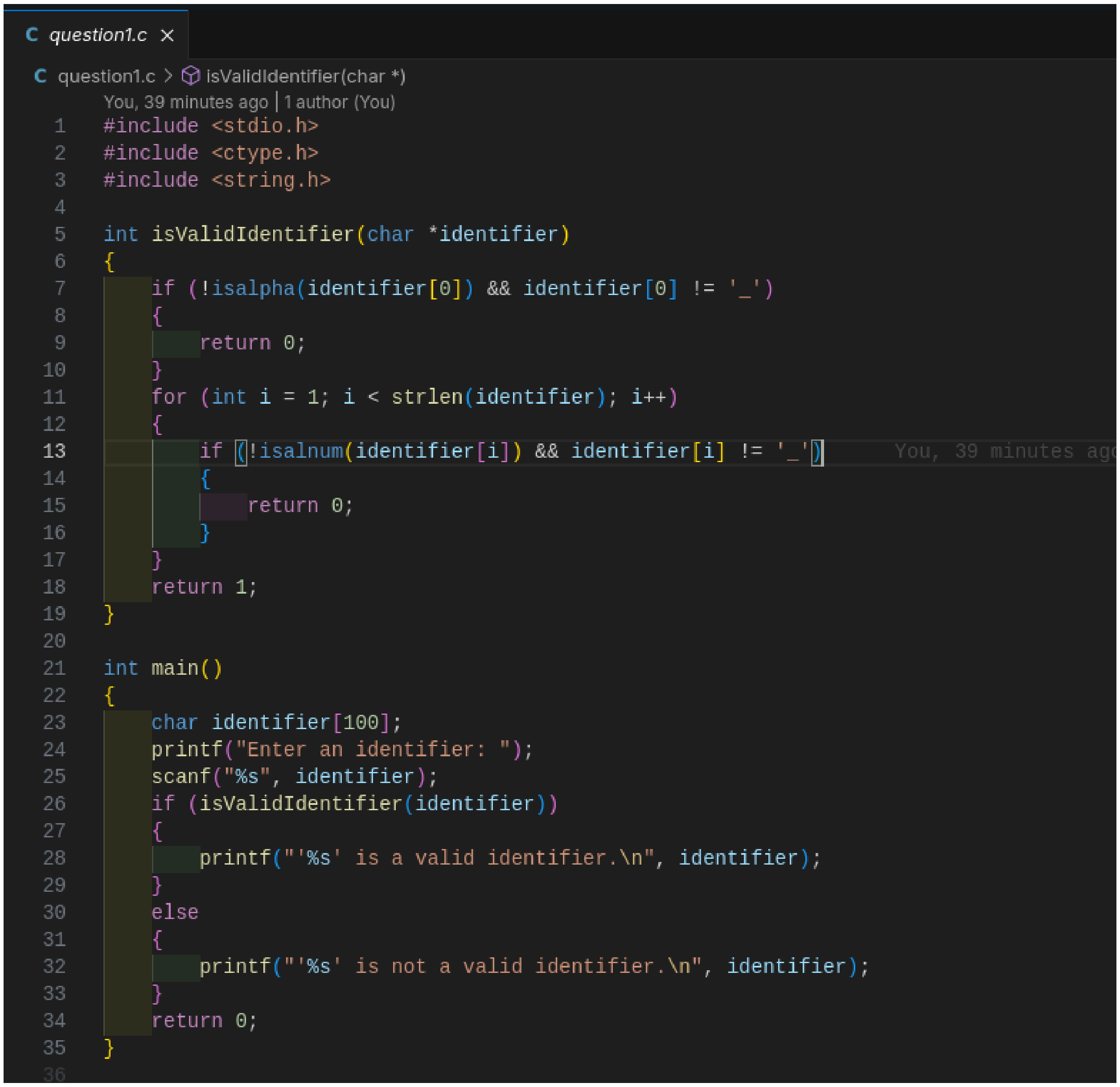
CS 358 LAB 1

Arnav Jain - 220002018

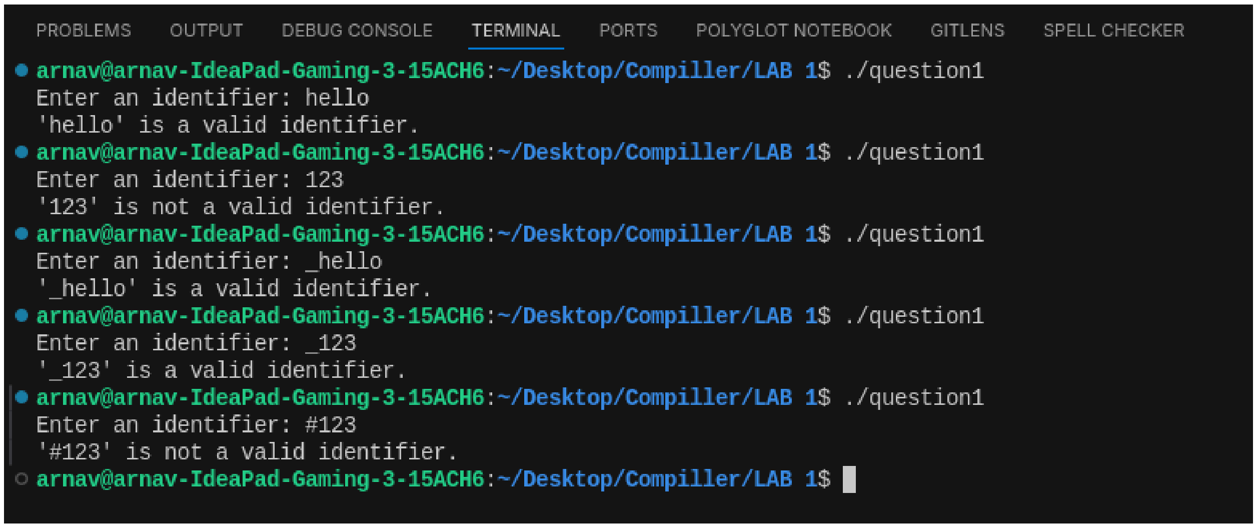
Question 1

Code:


question1.c
question1.c > isValidIdentifier(char *)
You, 39 minutes ago | 1 author (You)

```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int isValidIdentifier(char *identifier)
{
    if (!isalpha(identifier[0]) && identifier[0] != '_')
    {
        return 0;
    }
    for (int i = 1; i < strlen(identifier); i++)
    {
        if (!isalnum(identifier[i]) && identifier[i] != '_')        You, 39 minutes ago
        {
            return 0;
        }
    }
    return 1;
}

int main()
{
    char identifier[100];
    printf("Enter an identifier: ");
    scanf("%s", identifier);
    if (isValidIdentifier(identifier))
    {
        printf("'%s' is a valid identifier.\n", identifier);
    }
    else
    {
        printf("'%s' is not a valid identifier.\n", identifier);
    }
    return 0;
}
```

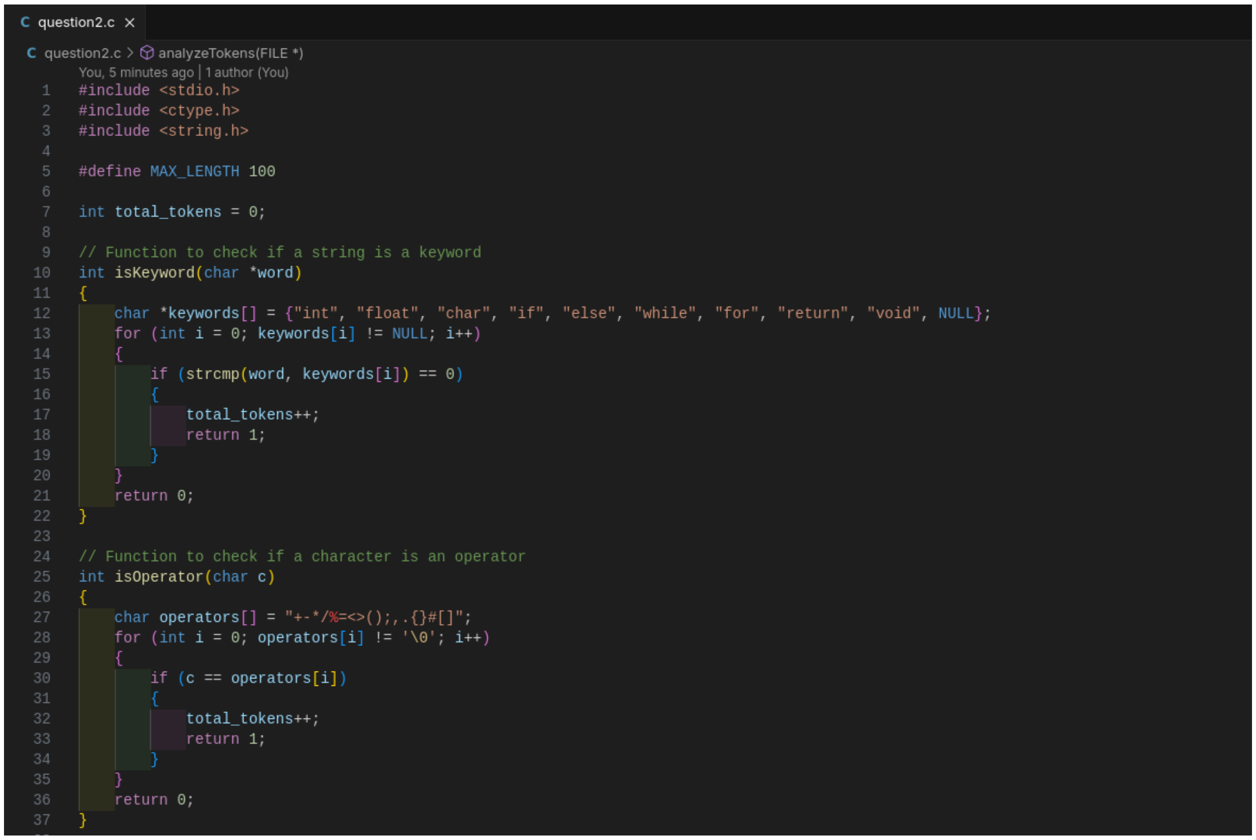
Output:



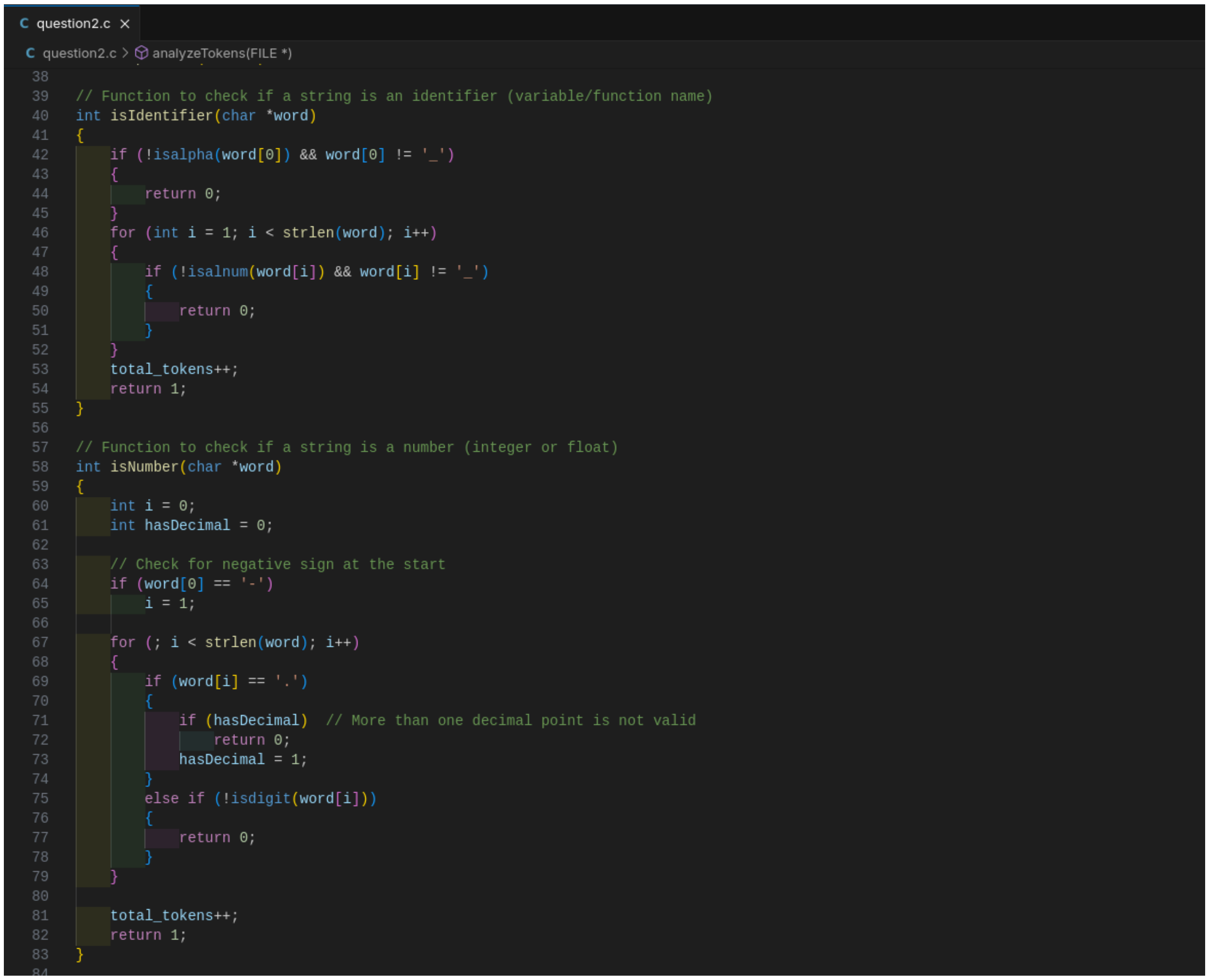
```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question1
Enter an identifier: hello
'hello' is a valid identifier.
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question1
Enter an identifier: 123
'123' is not a valid identifier.
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question1
Enter an identifier: _hello
'_hello' is a valid identifier.
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question1
Enter an identifier: _123
'_123' is a valid identifier.
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question1
Enter an identifier: #123
'#123' is not a valid identifier.
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$
```

Question 2

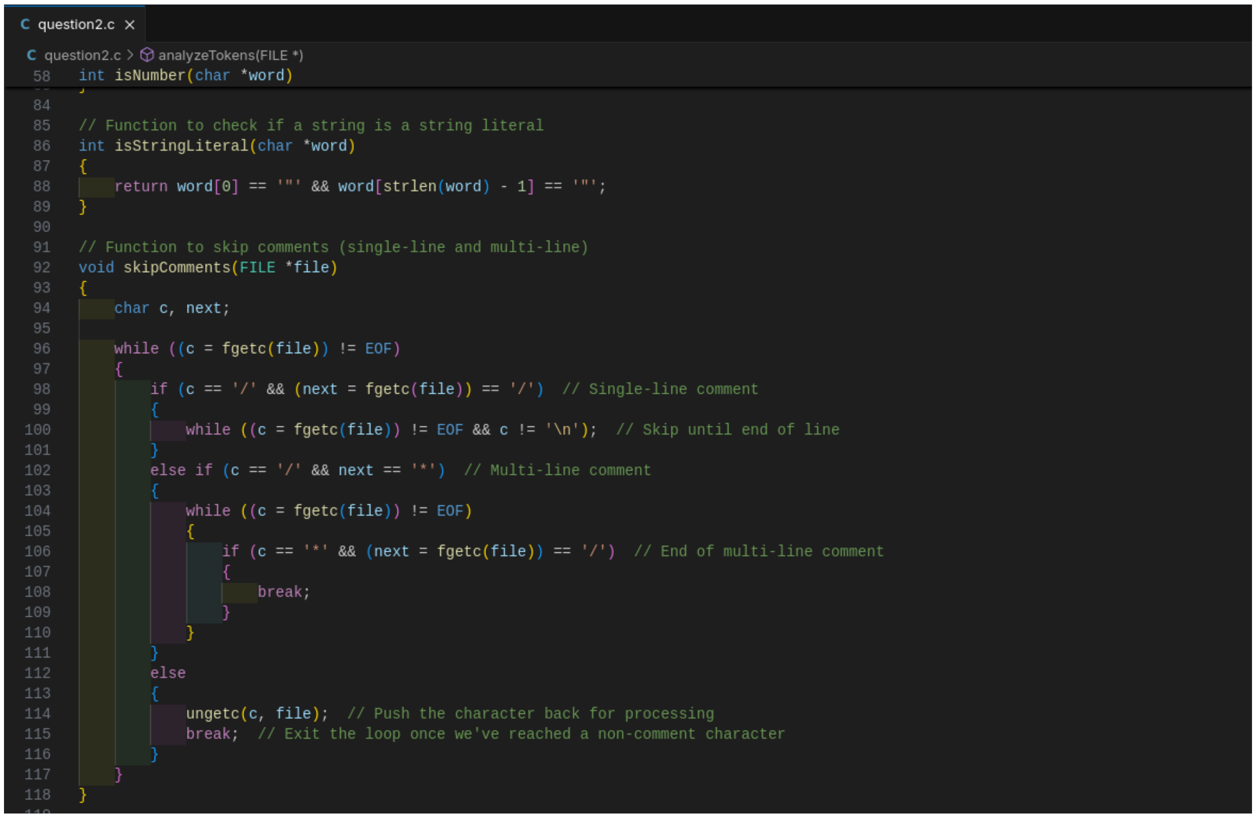
Code:


question2.c
question2.c > analyzeTokens(FILE *)
You, 5 minutes ago | 1 author (You)

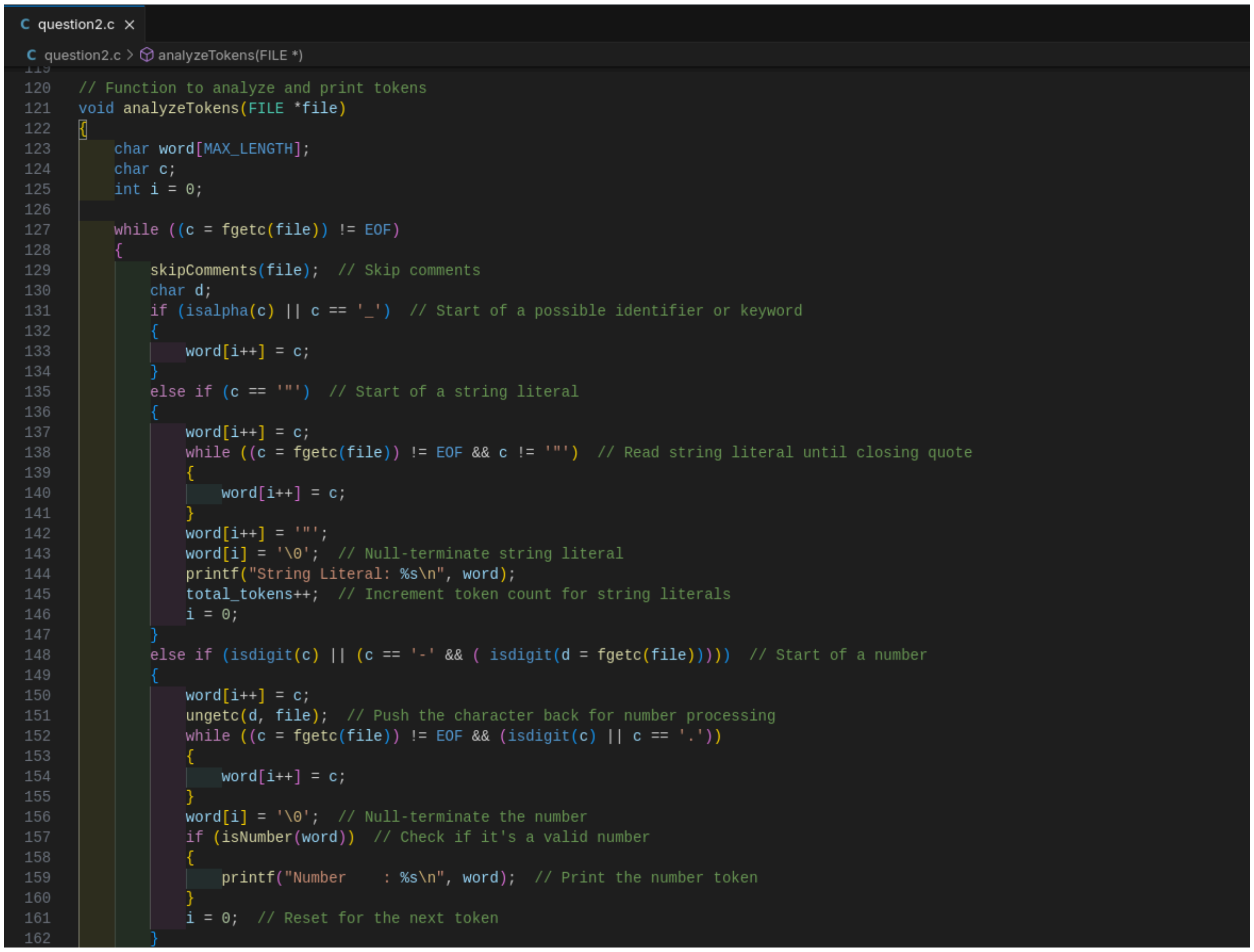
```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>

#define MAX_LENGTH 100

int total_tokens = 0;

// Function to check if a string is a keyword
int isKeyword(char *word)
{
    char *keywords[] = {"int", "float", "char", "if", "else", "while", "for", "return", "void", NULL};
    for (int i = 0; keywords[i] != NULL; i++)
    {
        if (strcmp(word, keywords[i]) == 0)
        {
            total_tokens++;
            return 1;
        }
    }
    return 0;
}

// Function to check if a character is an operator
int isOperator(char c)
{
    char operators[] = "+-*/%=<>();,.{}#[]";
    for (int i = 0; operators[i] != '\0'; i++)
    {
        if (c == operators[i])
        {
            total_tokens++;
            return 1;
        }
    }
    return 0;
}
```



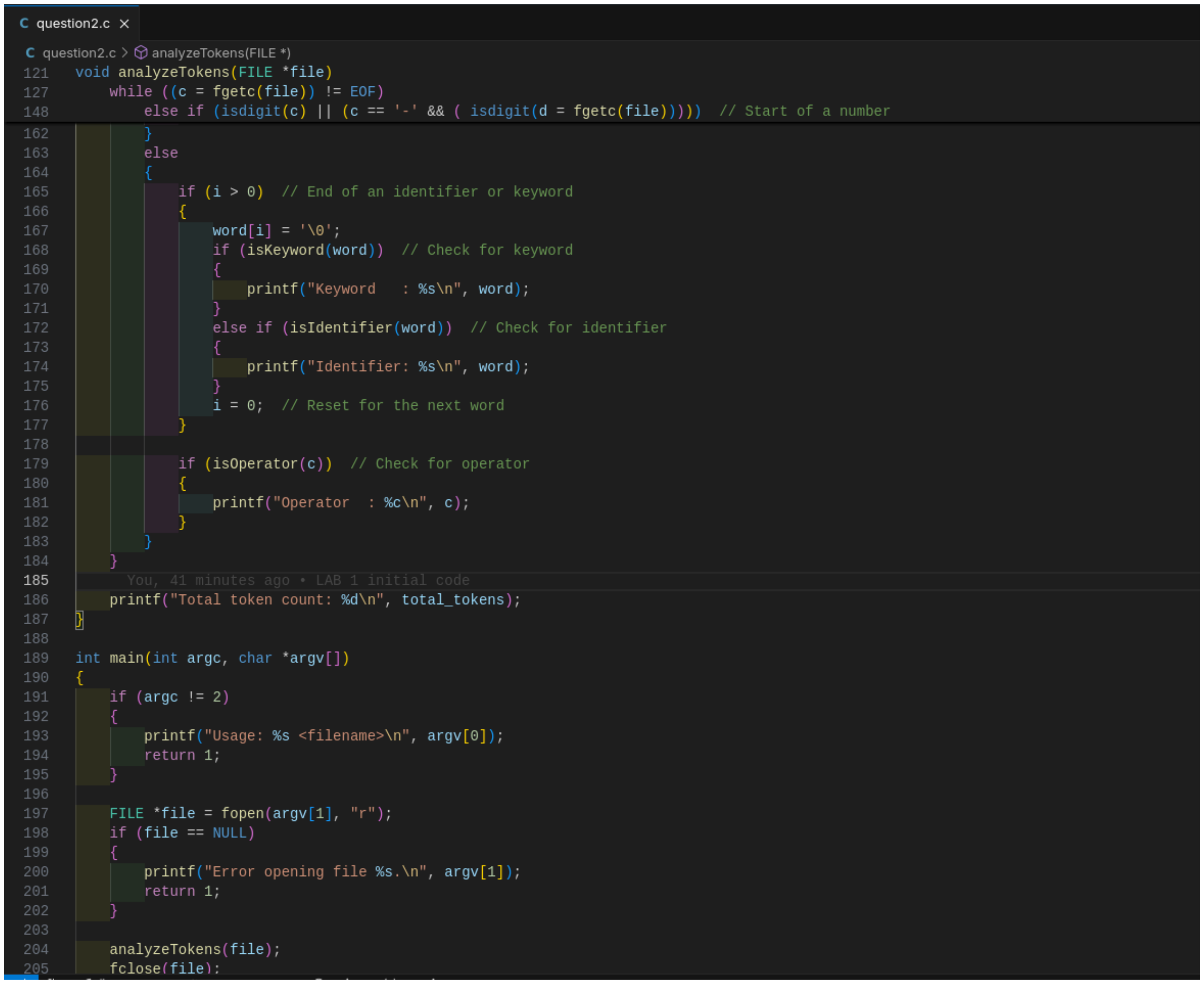
```c

// Function to check if a string is an identifier (variable/function name)
int isIdentifier(char *word)
{
    if (!isalpha(word[0]) && word[0] != '_')
    {
        return 0;
    }
    for (int i = 1; i < strlen(word); i++)
    {
        if (!isalnum(word[i]) && word[i] != '_')
        {
            return 0;
        }
    }
    total_tokens++;
    return 1;
}

// Function to check if a string is a number (integer or float)
int isNumber(char *word)
{
    int i = 0;
    int hasDecimal = 0;

    // Check for negative sign at the start
    if (word[0] == '-')
        i = 1;

    for (; i < strlen(word); i++)
    {
        if (word[i] == '.')
        {
            if (hasDecimal)  // More than one decimal point is not valid
                return 0;
            hasDecimal = 1;
        }
        else if (!isdigit(word[i]))
        {
            return 0;
        }
    }

    total_tokens++;
    return 1;
}
```



```c
int isNumber(char *word)

// Function to check if a string is a string literal
int isStringLiteral(char *word)
{
    return word[0] == '"' && word[strlen(word) - 1] == '"';
}

// Function to skip comments (single-line and multi-line)
void skipComments(FILE *file)
{
    char c, next;

    while ((c = fgetc(file)) != EOF)
    {
        if (c == '/' && (next = fgetc(file)) == '/')  // Single-line comment
        {
            while ((c = fgetc(file)) != EOF && c != '\n');  // Skip until end of line
        }
        else if (c == '/' && next == '*')  // Multi-line comment
        {
            while ((c = fgetc(file)) != EOF)
            {
                if (c == '*' && (next = fgetc(file)) == '/')  // End of multi-line comment
                {
                    break;
                }
            }
        }
        else
        {
            ungetc(c, file);  // Push the character back for processing
            break;  // Exit the loop once we've reached a non-comment character
        }
    }
}
```

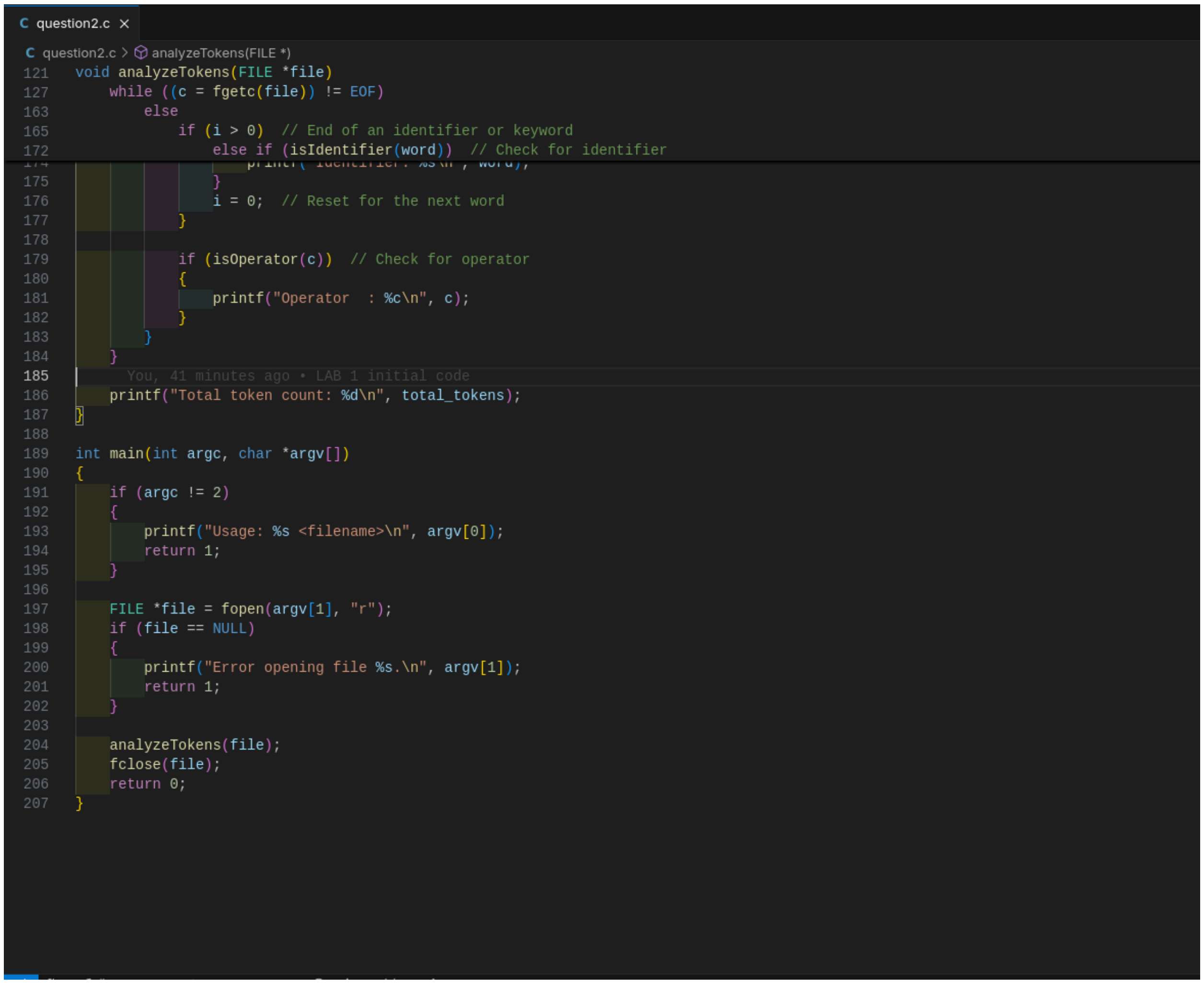


```c
// Function to analyze and print tokens
void analyzeTokens(FILE *file)
{
    char word[MAX_LENGTH];
    char c;
    int i = 0;

    while ((c = fgetc(file)) != EOF)
    {
        skipComments(file);  // Skip comments
        char d;
        if (isalpha(c) || c == '_')  // Start of a possible identifier or keyword
        {
            word[i++] = c;
        }
        else if (c == '"')  // Start of a string literal
        {
            word[i++] = c;
            while ((c = fgetc(file)) != EOF && c != '"')  // Read string literal until closing quote
            {
                word[i++] = c;
            }
            word[i++] = '"';
            word[i] = '\0';  // Null-terminate string literal
            printf("String Literal: %s\n", word);
            total_tokens++;  // Increment token count for string literals
            i = 0;
        }
        else if (isdigit(c) || (c == '-' && ( isdigit(d = fgetc(file)))))  // Start of a number
        {
            word[i++] = c;
            ungetc(d, file);  // Push the character back for number processing
            while ((c = fgetc(file)) != EOF && (isdigit(c) || c == '.'))
            {
                word[i++] = c;
            }
            word[i] = '\0';  // Null-terminate the number
            if (isNumber(word))  // Check if it's a valid number
            {
                printf("Number    : %s\n", word);  // Print the number token
            }
            i = 0;  // Reset for the next token
        }
```



question2.c
question2.c > analyzeTokens(FILE *)

```c
void analyzeTokens(FILE *file)
    while ((c = fgetc(file)) != EOF)
        else if (isdigit(c) || (c == '-' && ( isdigit(d = fgetc(file)))))  // Start of a number
        }
        else
        {
            if (i > 0)  // End of an identifier or keyword
            {
                word[i] = '\0';
                if (isKeyword(word))  // Check for keyword
                {
                    printf("Keyword    : %s\n", word);
                }
                else if (isIdentifier(word))  // Check for identifier
                {
                    printf("Identifier: %s\n", word);
                }
                i = 0;  // Reset for the next word
            }

            if (isOperator(c))  // Check for operator
            {
                printf("Operator   : %c\n", c);
            }
        }
    }
      You, 41 minutes ago • LAB 1 initial code
    printf("Total token count: %d\n", total_tokens);
}

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL)
    {
        printf("Error opening file %s.\n", argv[1]);
        return 1;
    }

    analyzeTokens(file);
    fclose(file);
```

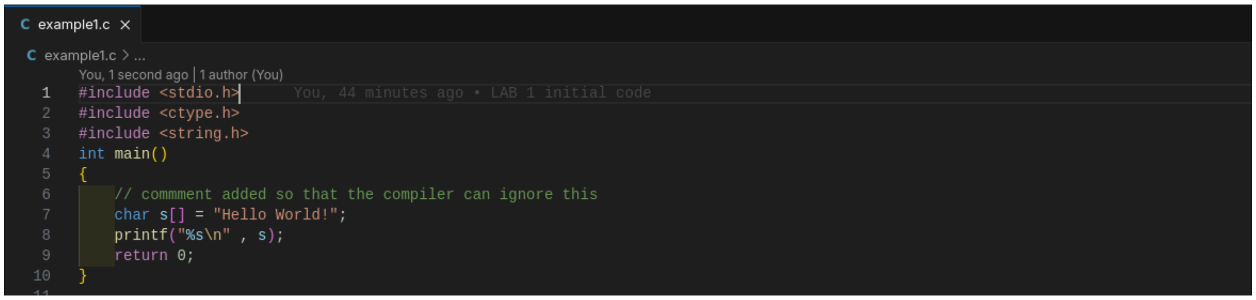


question2.c ×

question2.c > analyzeTokens(FILE *)

```c
void analyzeTokens(FILE *file)
    while ((c = fgetc(file)) != EOF)
        else
            if (i > 0)  // End of an identifier or keyword
                else if (isIdentifier(word))  // Check for identifier
                }
                i = 0;  // Reset for the next word
            }

            if (isOperator(c))  // Check for operator
            {
                printf("Operator  : %c\n", c);
            }
        }
    }
      You, 41 minutes ago • LAB 1 initial code
    printf("Total token count: %d\n", total_tokens);
}

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }

    FILE *file = fopen(argv[1], "r");
    if (file == NULL)
    {
        printf("Error opening file %s.\n", argv[1]);
        return 1;
    }

    analyzeTokens(file);
    fclose(file);
    return 0;
}
```

Examples on which this code was tested:

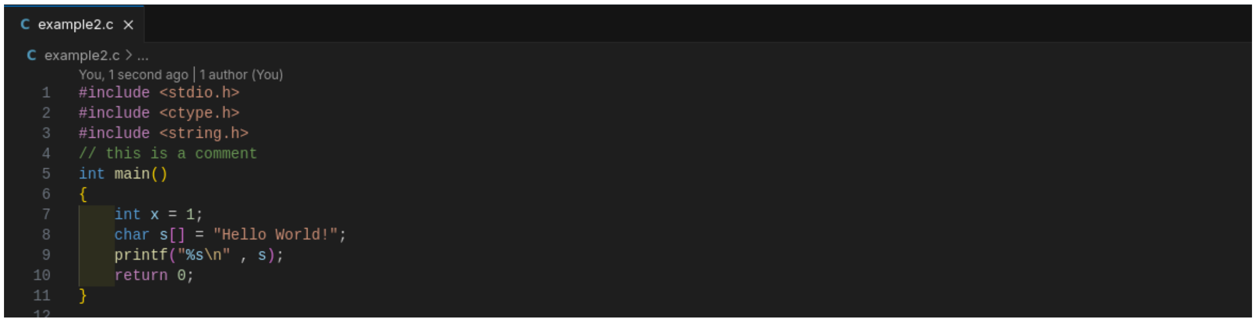


example1.c ×

example1.c > ...

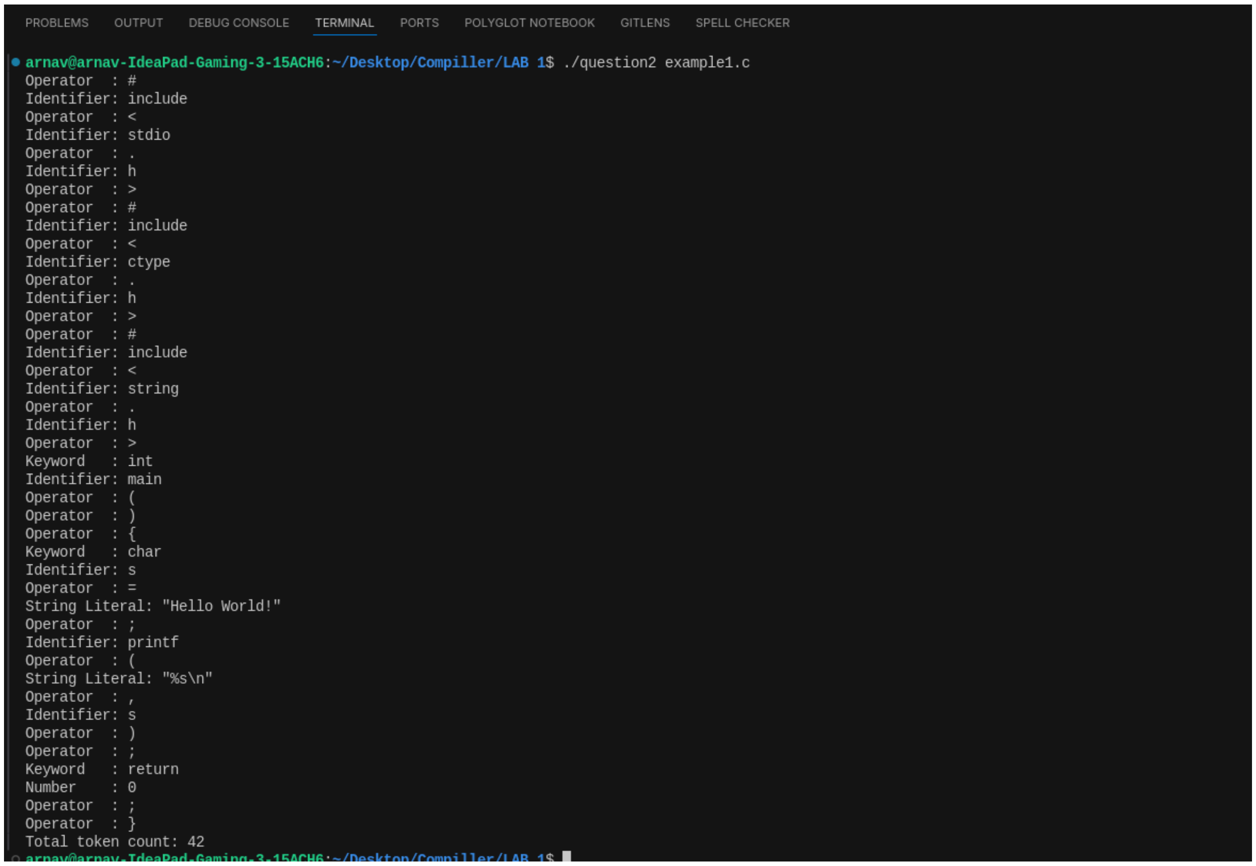
You, 1 second ago | 1 author (You)

```c
#include <stdio.h>      You, 44 minutes ago • LAB 1 initial code
#include <ctype.h>
#include <string.h>
int main()
{
    // commment added so that the compiler can ignore this
    char s[] = "Hello World!";
    printf("%s\n" , s);
    return 0;
}
```

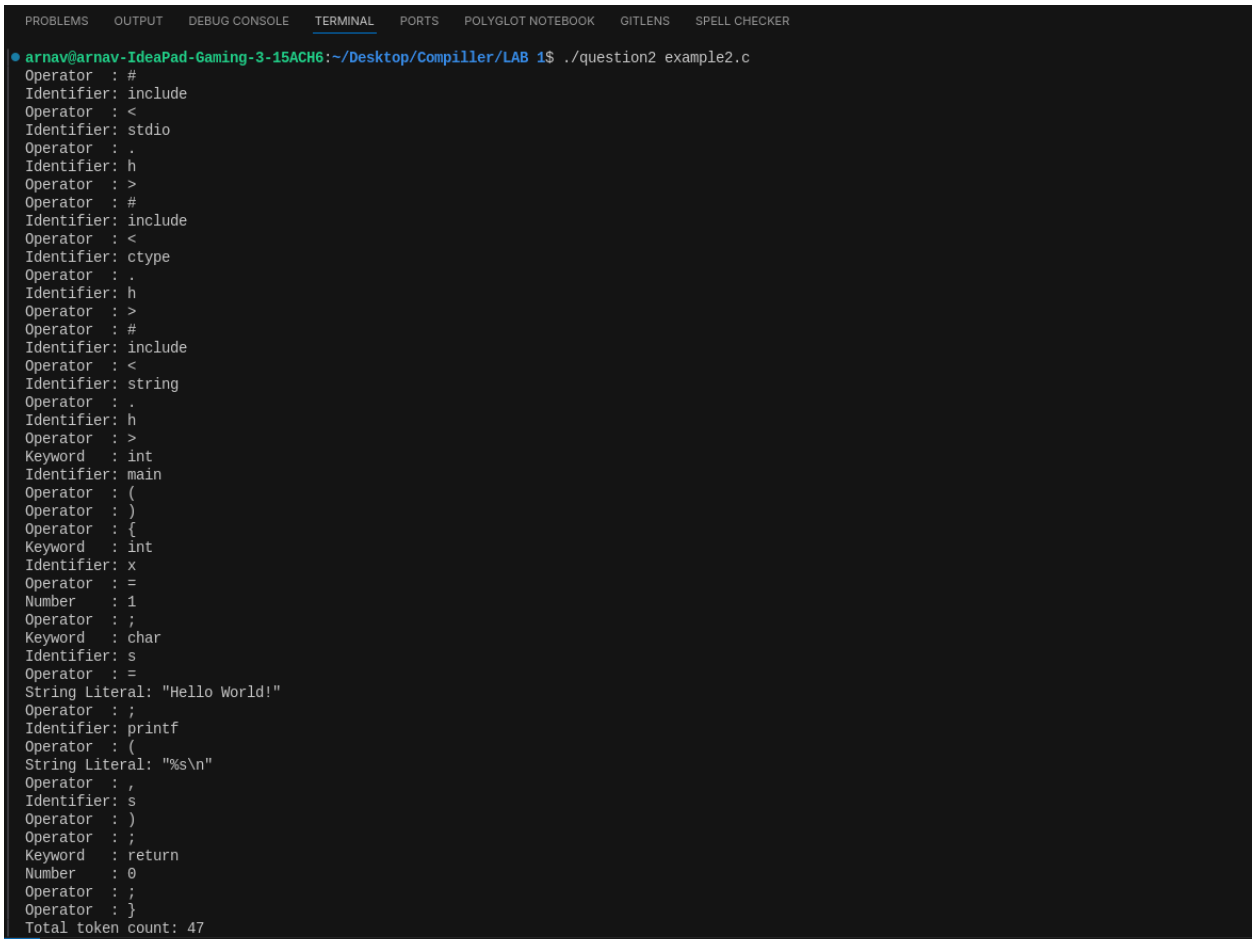


```c
#include <stdio.h>
#include <ctype.h>
#include <string.h>
// this is a comment
int main()
{
    int x = 1;
    char s[] = "Hello World!";
    printf("%s\n" , s);
    return 0;
}
```

Result:



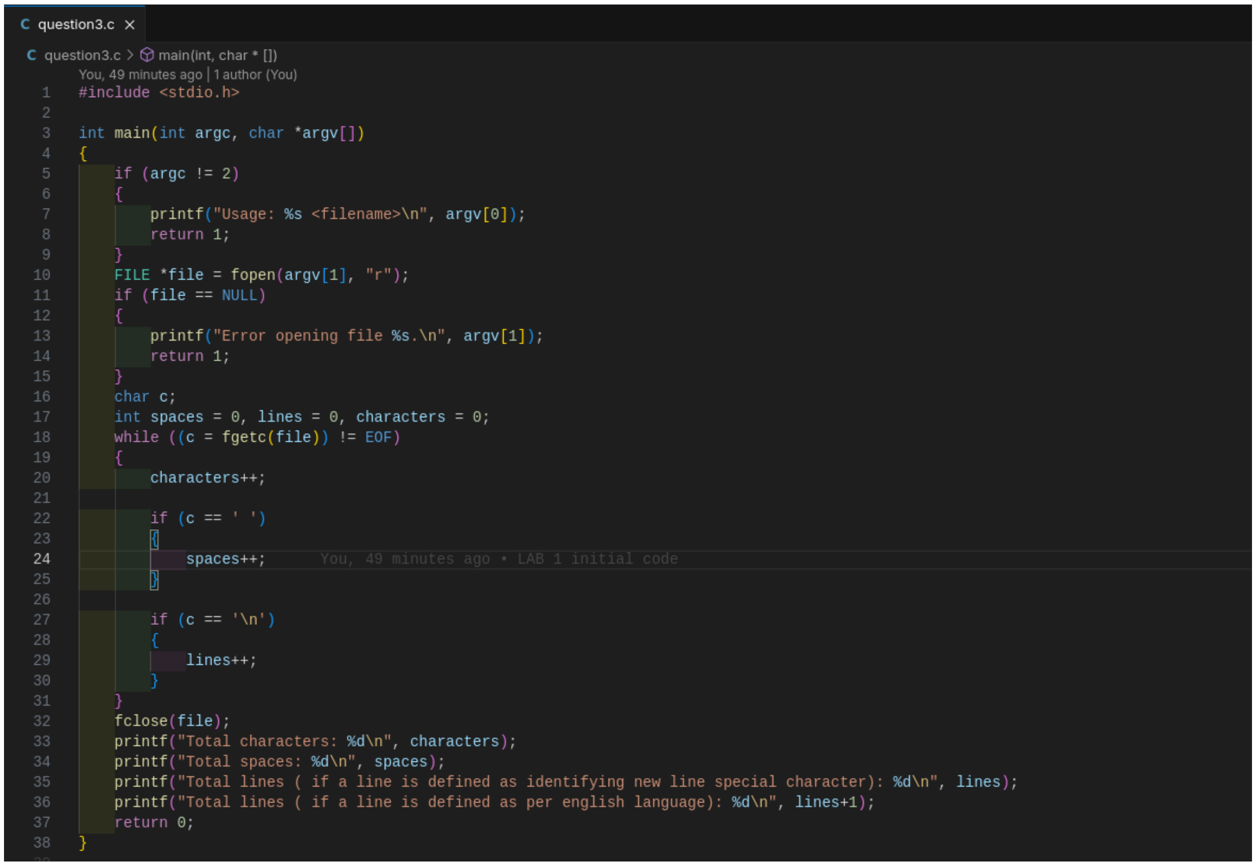
```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question2 example1.c
Operator  : #
Identifier: include
Operator  : <
Identifier: stdio
Operator  : .
Identifier: h
Operator  : >
Operator  : #
Identifier: include
Operator  : <
Identifier: ctype
Operator  : .
Identifier: h
Operator  : >
Operator  : #
Identifier: include
Operator  : <
Identifier: string
Operator  : .
Identifier: h
Operator  : >
Keyword   : int
Identifier: main
Operator  : (
Operator  : )
Operator  : {
Keyword   : char
Identifier: s
Operator  : =
String Literal: "Hello World!"
Operator  : ;
Identifier: printf
Operator  : (
String Literal: "%s\n"
Operator  : ,
Identifier: s
Operator  : )
Operator  : ;
Keyword   : return
Number    : 0
Operator  : ;
Operator  : }
Total token count: 42
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$
```



```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question2 example2.c
Operator  : #
Identifier: include
Operator  : <
Identifier: stdio
Operator  : .
Identifier: h
Operator  : >
Operator  : #
Identifier: include
Operator  : <
Identifier: ctype
Operator  : .
Identifier: h
Operator  : >
Operator  : #
Identifier: include
Operator  : <
Identifier: string
Operator  : .
Identifier: h
Operator  : >
Keyword   : int
Identifier: main
Operator  : (
Operator  : )
Operator  : {
Keyword   : int
Identifier: x
Operator  : =
Number    : 1
Operator  : ;
Keyword   : char
Identifier: s
Operator  : =
String Literal: "Hello World!"
Operator  : ;
Identifier: printf
Operator  : (
String Literal: "%s\n"
Operator  : ,
Identifier: s
Operator  : )
Operator  : ;
Keyword   : return
Number    : 0
Operator  : ;
Operator  : }
Total token count: 47
```

Question 3

Code:

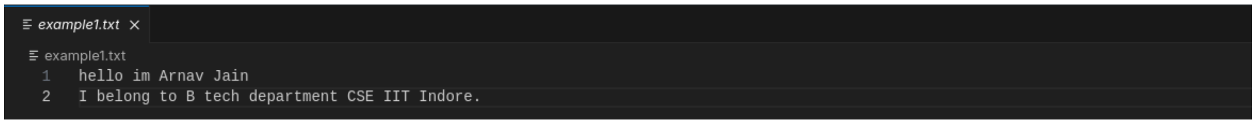

question3.c

question3.c > main(int, char * [])

You, 49 minutes ago | 1 author (You)

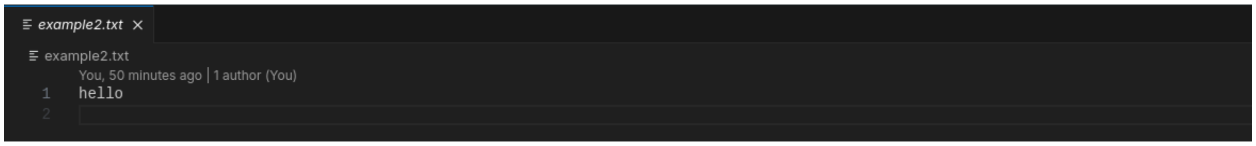
```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    if (argc != 2)
    {
        printf("Usage: %s <filename>\n", argv[0]);
        return 1;
    }
    FILE *file = fopen(argv[1], "r");
    if (file == NULL)
    {
        printf("Error opening file %s.\n", argv[1]);
        return 1;
    }
    char c;
    int spaces = 0, lines = 0, characters = 0;
    while ((c = fgetc(file)) != EOF)
    {
        characters++;

        if (c == ' ')
        {
            spaces++;       You, 49 minutes ago • LAB 1 initial code
        }

        if (c == '\n')
        {
            lines++;
        }
    }
    fclose(file);
    printf("Total characters: %d\n", characters);
    printf("Total spaces: %d\n", spaces);
    printf("Total lines ( if a line is defined as identifying new line special character): %d\n", lines);
    printf("Total lines ( if a line is defined as per english language): %d\n", lines+1);
    return 0;
}
```

Examples on which this code was tested on:


example1.txt

example1.txt

```
hello im Arnav Jain
I belong to B tech department CSE IIT Indore.
```

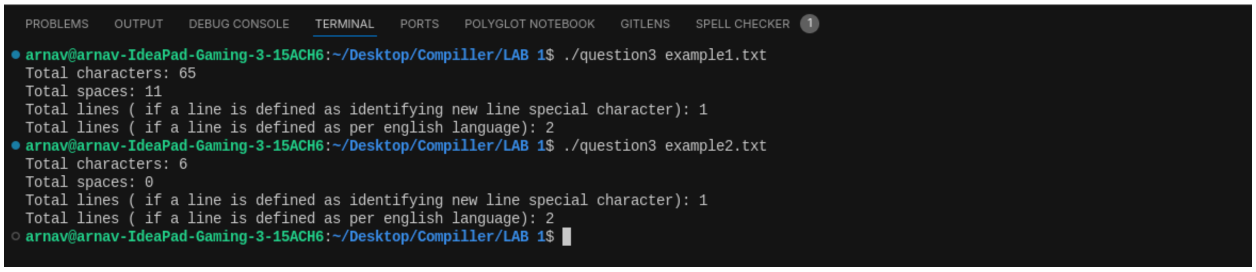

example2.txt

example2.txt

You, 50 minutes ago | 1 author (You)

```
hello

```

Result:


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POLYGLOT NOTEBOOK GITLENS SPELL CHECKER 1

```
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question3 example1.txt
Total characters: 65
Total spaces: 11
Total lines ( if a line is defined as identifying new line special character): 1
Total lines ( if a line is defined as per english language): 2
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ ./question3 example2.txt
Total characters: 6
Total spaces: 0
Total lines ( if a line is defined as identifying new line special character): 1
Total lines ( if a line is defined as per english language): 2
arnav@arnav-IdeaPad-Gaming-3-15ACH6:~/Desktop/Compiller/LAB 1$ 
```

To view the code files , refer github

https://github.com/arnavjain2710/Compiller-Techniques/tree/main/LAB%201