LAB REPORT

CSE332: Compiler Design Lab

03

Topic: Solving Problem Using C.

Submitted To

Shadman Rabby (SHR)

Lecturer

Department of CSE,

Daffodil International University

Submitted By

Student ID: 221-15-5400

Section: 61_A2

Student Name: M. B. Mahir Tanzim

Date of Assignment Submission: 27 March 2025

Experiment No: 03		**Mapping:** CO1 and CO2	
Experiment Name	Solving Problem Using C		

Experiment Details:

Problem 01: Write a C program that will take multiple lines as input and count the number of Lines.

Solution:

```
#include<bits/stdc++.h>

using namespace std;
int main() {
    string line;
    int count = 0;
    while (getline(cin, line)) {
        count++;
    }
    cout << "Total number of lines: " << count << endl;
    return 0;
}
```

Problem 02: Write a C program that will take multiple lines as input and identify the comments if there any.

Solution

```
#include <stdio.h>
#include <string.h>
void detect_comments(char *line){
    if (strstr(line, "//") != NULL){
        printf("Single-line comment detected: %s", line);
    }
    else if (strstr(line, "/*") != NULL){
        printf("Multi-line comment detected: %s", line);
    }
```

```c
}
int main(){
    char line[1000];
    printf("Enter the C code (Press Ctrl+Z to stop input):\n");
    while (fgets(line, sizeof(line), stdin) != NULL){
        detect_comments(line);
    }
    return 0;
}
```

Problem 03: Write a C program that will take multiple lines as input and remove the single line/multiple line comments if there any.

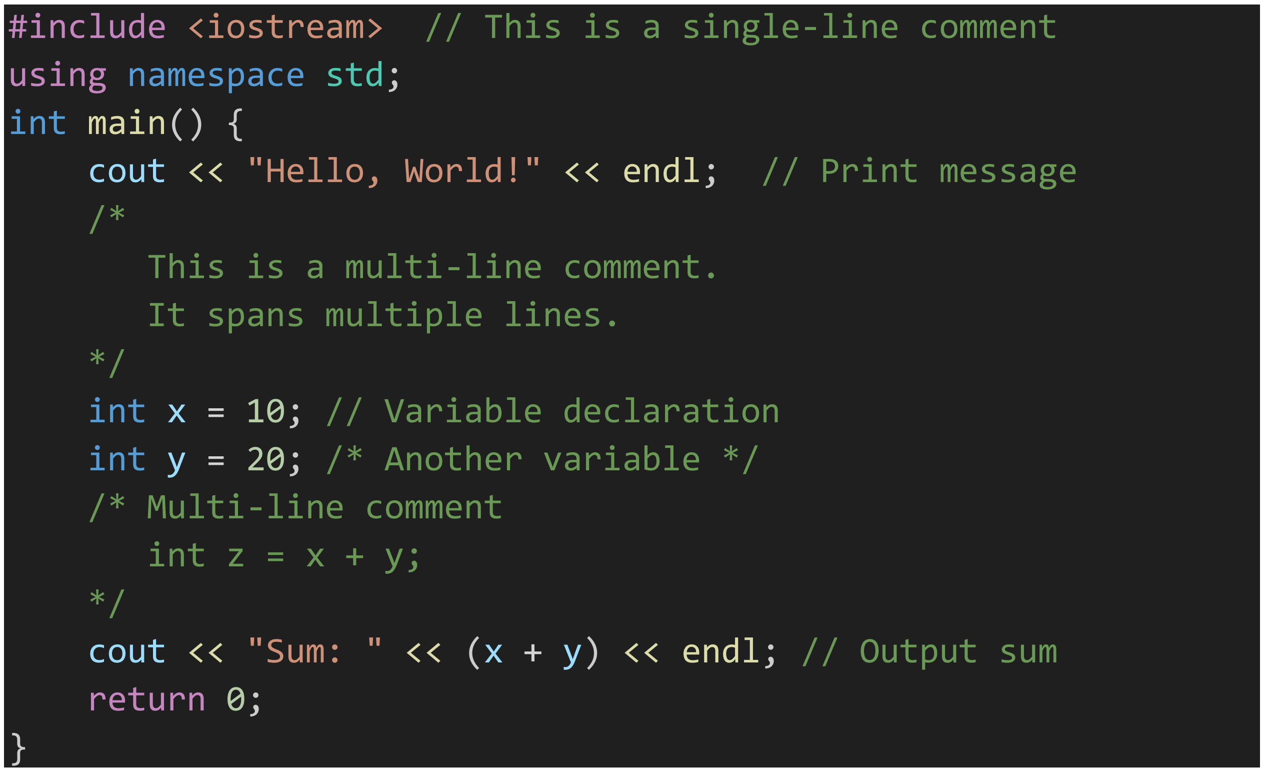
Solution:

```c
#include <stdio.h>
#include <string.h>
void remove_comments(char *code) {
    int i = 0, j = 0;
    int in_string = 0, in_char = 0;
    int in_single_comment = 0, in_multi_comment = 0;
    char result[1000];
    while (code[i]) {
        if (in_single_comment && code[i] == '\n') {
            in_single_comment = 0;
            result[j++] = code[i];
        }
        else if (in_multi_comment && code[i] == '*' && code[i +
1] == '/') {
            in_multi_comment = 0;
            i++;
        }
        else if (in_single_comment || in_multi_comment) {
            // Skip characters inside comments
        }
```

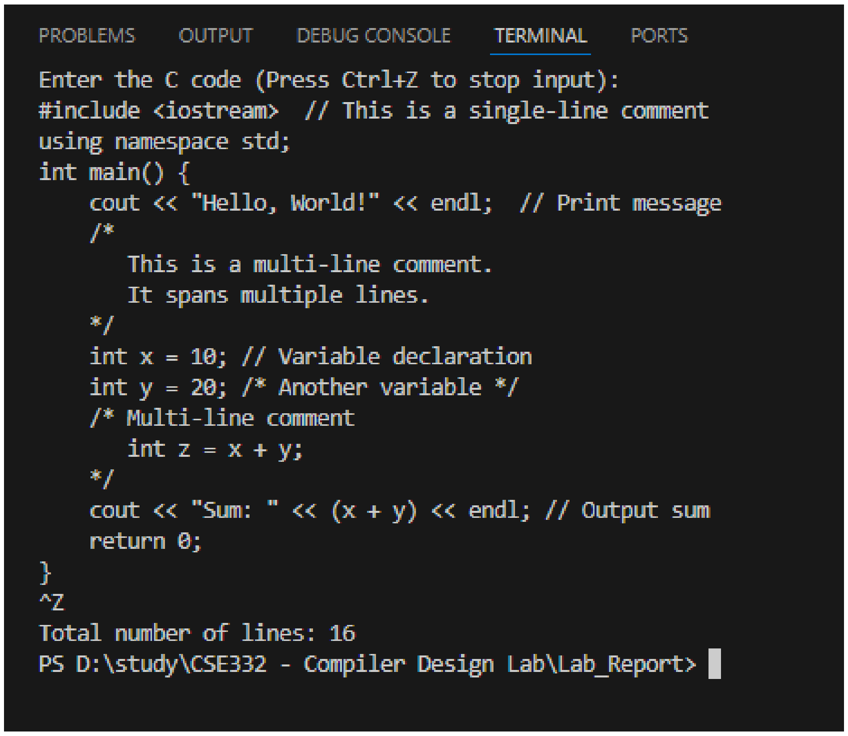
```
         else if (code[i] == '/' && code[i + 1] == '/') {
            in_single_comment = 1;
        }
         else if (code[i] == '/' && code[i + 1] == '*') {
            in_multi_comment = 1;
            i++;
        }
        else {
            result[j++] = code[i];
        }
        i++;
    }
    result[j] = '\0';
    printf("Code without comments:\n%s", result);
}
int main() {
    char code[1000], line[200];

    printf("Enter the C code (Press Ctrl+D to stop input on Linux/Mac, Ctrl+Z on Windows):\n");

    code[0] = '\0';
    while (fgets(line, sizeof(line), stdin) != NULL) {
        strcat(code, line);
    }
    remove_comments(code);
    return 0;
}
```

Obtained Output:

After taking the following demo code as input in every Solution, we obtain the desired output:

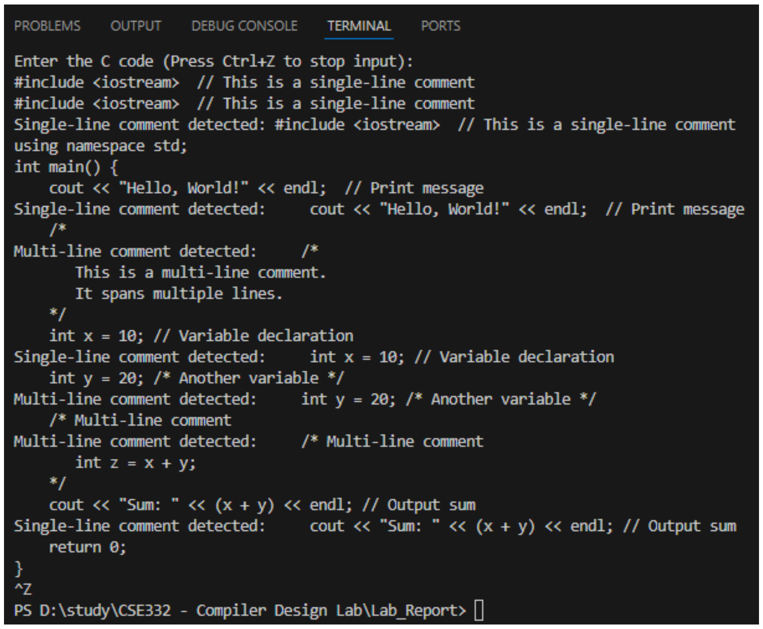


```
#include <iostream>  // This is a single-line comment
using namespace std;
int main() {
    cout << "Hello, World!" << endl;  // Print message
    /*
       This is a multi-line comment.
       It spans multiple lines.
    */
    int x = 10; // Variable declaration
    int y = 20; /* Another variable */
    /* Multi-line comment
       int z = x + y;
    */
    cout << "Sum: " << (x + y) << endl; // Output sum
    return 0;
}
```

Problem 01:	Desired Output?
	YES

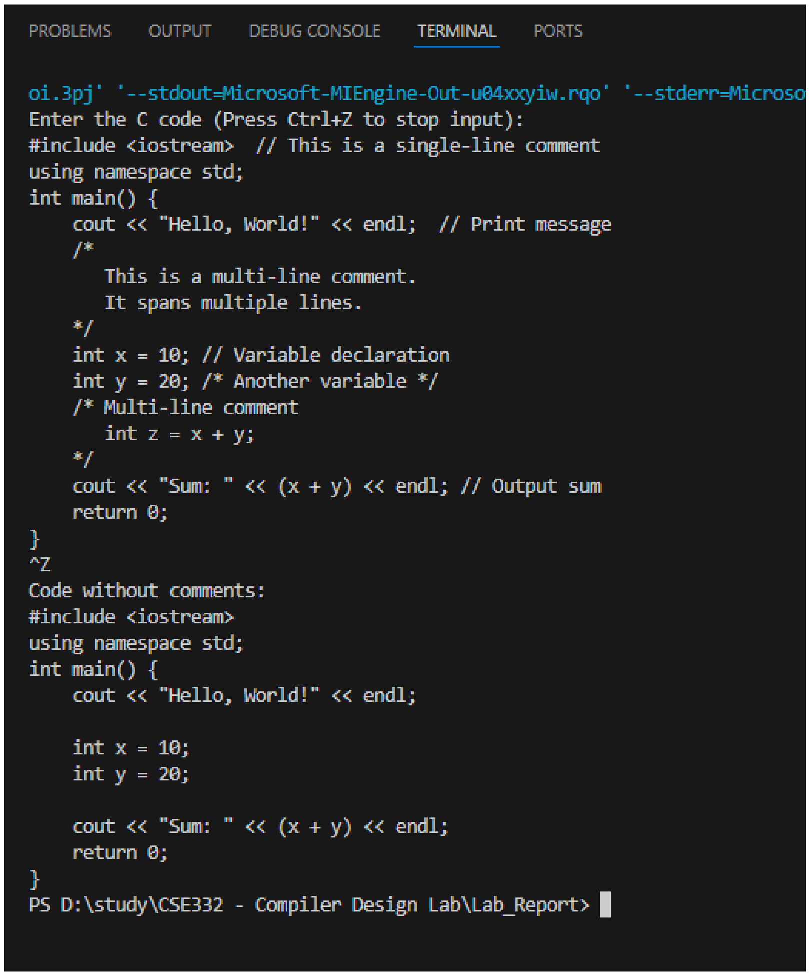
```
Enter the C code (Press Ctrl+Z to stop input):
#include <iostream>  // This is a single-line comment
using namespace std;
int main() {
    cout << "Hello, World!" << endl;  // Print message
    /*
       This is a multi-line comment.
       It spans multiple lines.
    */
    int x = 10; // Variable declaration
    int y = 20; /* Another variable */
    /* Multi-line comment
       int z = x + y;
    */
    cout << "Sum: " << (x + y) << endl; // Output sum
    return 0;
}
^Z
Total number of lines: 16
PS D:\study\CSE332 - Compiler Design Lab\Lab_Report>
```

Problem 02:



```
Enter the C code (Press Ctrl+Z to stop input):
#include <iostream>  // This is a single-line comment
#include <iostream>  // This is a single-line comment
Single-line comment detected: #include <iostream>  // This is a single-line comment
using namespace std;
int main() {
    cout << "Hello, World!" << endl;  // Print message
Single-line comment detected:     cout << "Hello, World!" << endl;  // Print message
    /*
Multi-line comment detected:     /*
       This is a multi-line comment.
       It spans multiple lines.
    */
    int x = 10; // Variable declaration
Single-line comment detected:     int x = 10; // Variable declaration
    int y = 20; /* Another variable */
Multi-line comment detected:     int y = 20; /* Another variable */
    /* Multi-line comment
Multi-line comment detected:     /* Multi-line comment
       int z = x + y;
    */
    cout << "Sum: " << (x + y) << endl; // Output sum
Single-line comment detected:     cout << "Sum: " << (x + y) << endl; // Output sum
    return 0;
}
^Z
PS D:\study\CSE332 - Compiler Design Lab\Lab_Report>
```

Problem 03:



```
oi.3pj' '--stdout=Microsoft-MIEngine-Out-u04xxyiw.rqo' '--stderr=Microso
Enter the C code (Press Ctrl+Z to stop input):
#include <iostream>  // This is a single-line comment
using namespace std;
int main() {
    cout << "Hello, World!" << endl;  // Print message
    /*
       This is a multi-line comment.
       It spans multiple lines.
    */
    int x = 10; // Variable declaration
    int y = 20; /* Another variable */
    /* Multi-line comment
       int z = x + y;
    */
    cout << "Sum: " << (x + y) << endl; // Output sum
    return 0;
}
^Z
Code without comments:
#include <iostream>
using namespace std;
int main() {
    cout << "Hello, World!" << endl;

    int x = 10;
    int y = 20;

    cout << "Sum: " << (x + y) << endl;
    return 0;
}
PS D:\study\CSE332 - Compiler Design Lab\Lab_Report>
```

YES

Conclusion:

In this lab, we successfully implemented programs to count the number of lines, detect comments, and remove comments from multiple lines of input. We explored different methods for each task and analyzed their advantages and limitations. The character-by-character approach proved to be efficient in memory usage, while line-by-line processing using fgets() provided better readability and ease of implementation.

For comment detection and removal, maintaining flags helped in accurately identifying single-line (//) and multi-line (/* ... */) comments. By applying structured logic, we ensured that comments were correctly removed while preserving the rest of the input.

Overall, this lab helped in understanding input processing techniques, string manipulation, and efficient handling of text-based data in C.