Department of Computer Science and Engineering
Premier University

CSE 454: Compiler Construction Laboratory

Lab Report 02: A program to identify whether a given line is a comment or not and generate a commentfree C program

Submitted by:

Name	Mohammad Hafizur Rahman Sakib
ID	0222210005101118
Section	C
Session	Fall 2025
Semester	8th Semester
Submission Date	31.01.2026

Submitted to:
Nazma Akther
Assistant Professor, Department of CSE
Premier University,Chattogram

Remarks

Experiment No 02

Experiment Name

A program to identify whether a given line is a comment or not and generate a comment-free C program.

Objective

The objective of this experiment is to design and implement a basic lexical analyzer to detect comments in a C program. The program identifies both single-line and multi-line comments, displays the line numbers where comments occur, and generates a new file containing the source code without comments. This experiment helps in understanding lexical analysis concepts used in compiler design.

Algorithm

1. Start the program.
2. Open the input C file in read mode and the output file in write mode.
3. Initialize variables to track line numbers and multi-line comments.
4. Read the source file line by line.
5. Identify single-line and multi-line comments.
6. Display the line numbers where comments are detected.
7. Write only non-comment code into the output file.
8. Repeat until the end of file is reached.
9. Close all files.
10. End the program.

Source Code

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

int main() {
    FILE *in, *out;
    char line[300];
    int i, lineNo = 0, block = 0;
    in = fopen("input.c", "r");
    out = fopen("output.c", "w");
    if (!in || !out) return 1;

    while (fgets(line, sizeof(line), in)) {
        lineNo++; i = 0;

        if (!block && strstr(line, "//"))
            printf("Single-line␣comment␣at␣line␣%d\n", lineNo);

        while (line[i]) {

            if (!block && line[i]=='/' && line[i+1]=='*')
                block = 1, printf("Block␣comment␣starts␣at␣line␣%
                   d\n", lineNo), i+=2;

            else if (block && line[i]=='*' && line[i+1]=='/')
                block = 0, printf("Block␣comment␣ends␣at␣line␣%d\
                   n", lineNo), i+=2;

            else if (block || (line[i]=='/' && line[i+1]=='/'))
                break;

            else
                fputc(line[i++], out);
        }
        if (!block) fputc('\n', out);
    }
    fclose(in); fclose(out);
    return 0;
}
```

Input

The input file `input.c` contains a C program with single-line and multi-line comments.

```
#include <stdio.h>

// This is a single-line comment
int main() {
    /* This is a
       multi-line comment */
    printf("Hello␣World");
    return 0;
}
```

Output

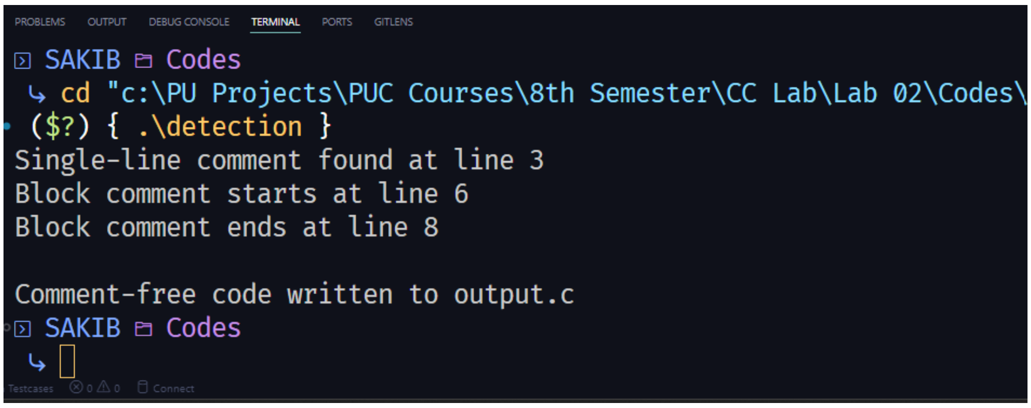
The output file `output.c` contains the same C program after removing all comments.

```
#include <stdio.h>

int main() {

    printf("Hello␣World");
    return 0;
}
```

Output Screenshot



```
SAKIB 🗀 Codes
 ↳ cd "c:\PU Projects\PUC Courses\8th Semester\CC Lab\Lab 02\Codes\
 ($?) { .\detection }
Single-line comment found at line 3
Block comment starts at line 6
Block comment ends at line 8

Comment-free code written to output.c
SAKIB 🗀 Codes
 ↳ 
```

Discussion

This experiment demonstrates the working of a basic lexical analyzer used in compiler design. The program successfully detects both single-line and multi-line comments and displays their respective line numbers. By removing comments and generating a clean output file, the experiment highlights the importance of lexical analysis in source code processing. File handling and string manipulation concepts in C were effectively applied.