Experiment No: 02

Experiment Name: Write a program to detect comment from a C program and remove it in the output file.

Objectives:

- To develop a program that detects and removes **single-line** (//) and **multi-line** (/*...*/) comments from a C source code file.
- To use **Python and Regular Expressions** (regex) to simulate the preprocessing step of a compiler.
- To count and display the total number of comments removed.
- To write the cleaned code into an output file for further compilation or analysis.

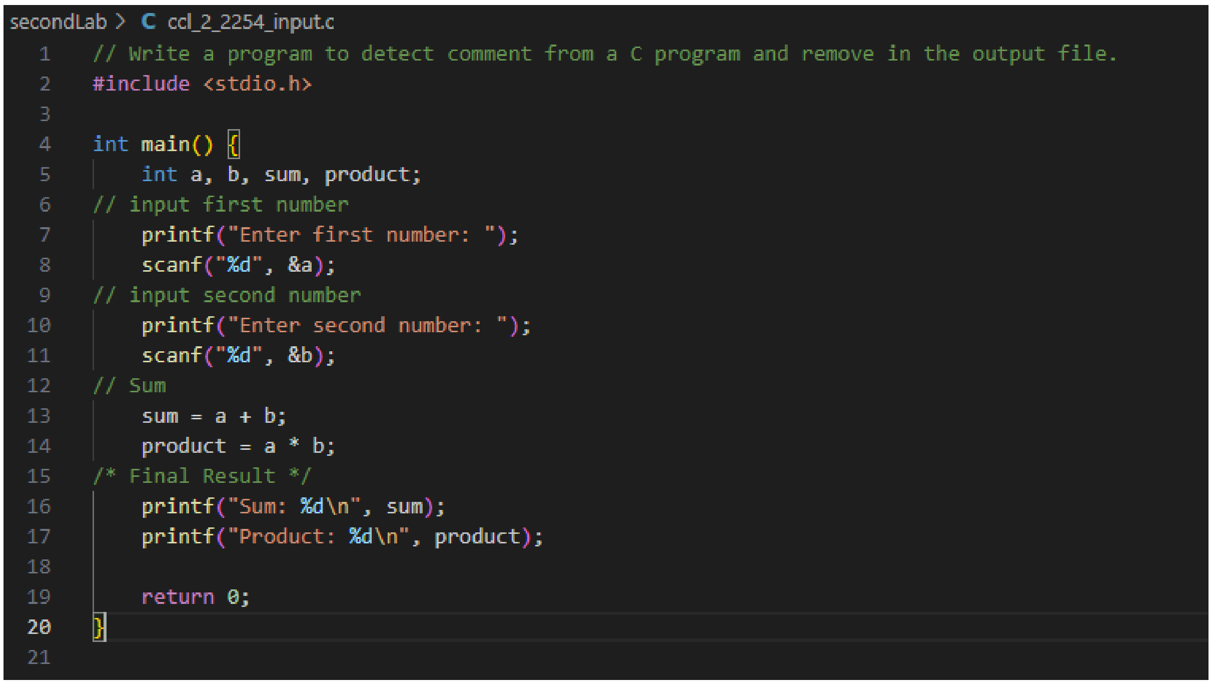
Algorithm:

<u>Start the program.</u>

1. Open the C source code (.c) file and read its contents into memory.
2. Use regular expressions to:
   - Find and count all // single-line comments.
   - Find and count all /* ... */ multi-line comments.
3. Remove all detected comments from the code.
4. Write the cleaned code into a new output file.
5. Print the count of removed single-line and multi-line comments.
6. End the program.

Code:

Input file:

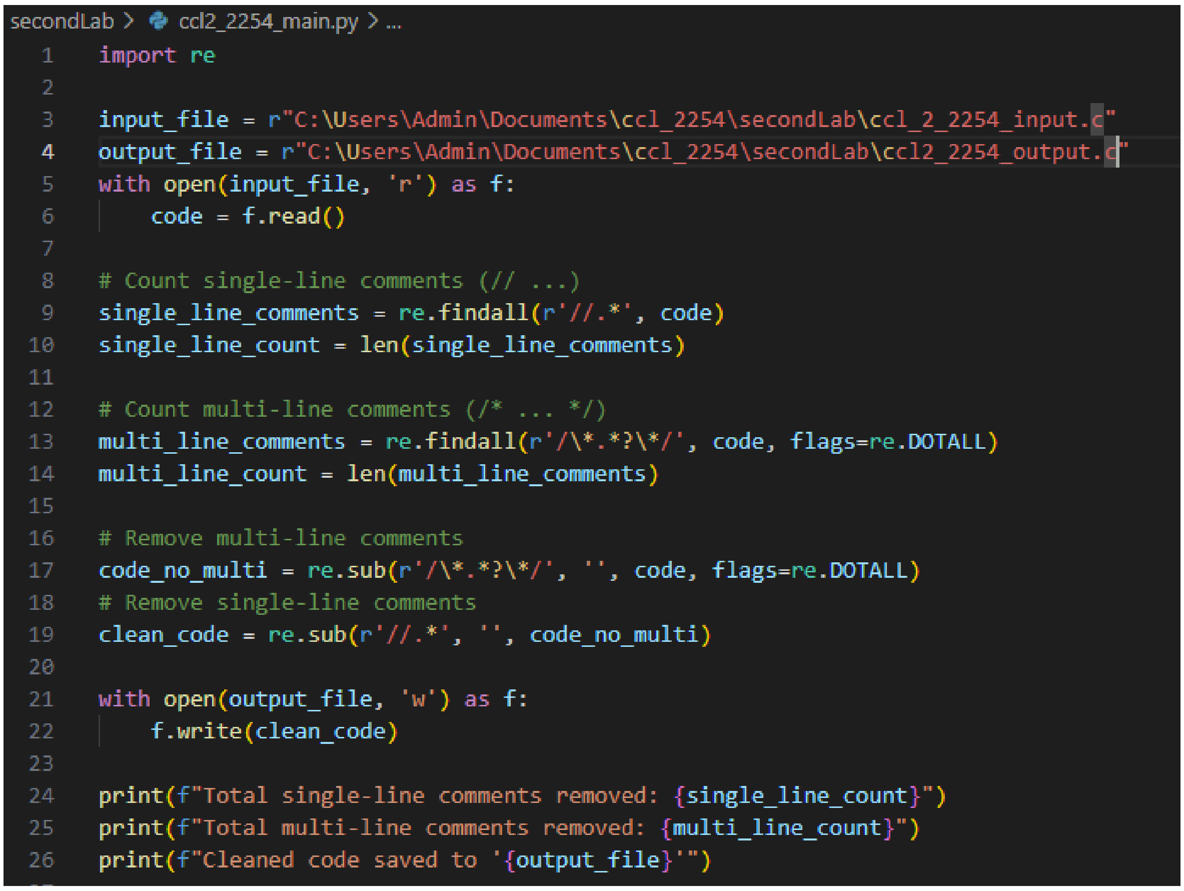


secondLab > C ccl_2_2254_input.c

```c
// Write a program to detect comment from a C program and remove in the output file.
#include <stdio.h>

int main() {
    int a, b, sum, product;
// input first number
    printf("Enter first number: ");
    scanf("%d", &a);
// input second number
    printf("Enter second number: ");
    scanf("%d", &b);
// Sum
    sum = a + b;
    product = a * b;
/* Final Result */
    printf("Sum: %d\n", sum);
    printf("Product: %d\n", product);

    return 0;
}

```

Figure01: ccl_2_2254_input.c

Main file:

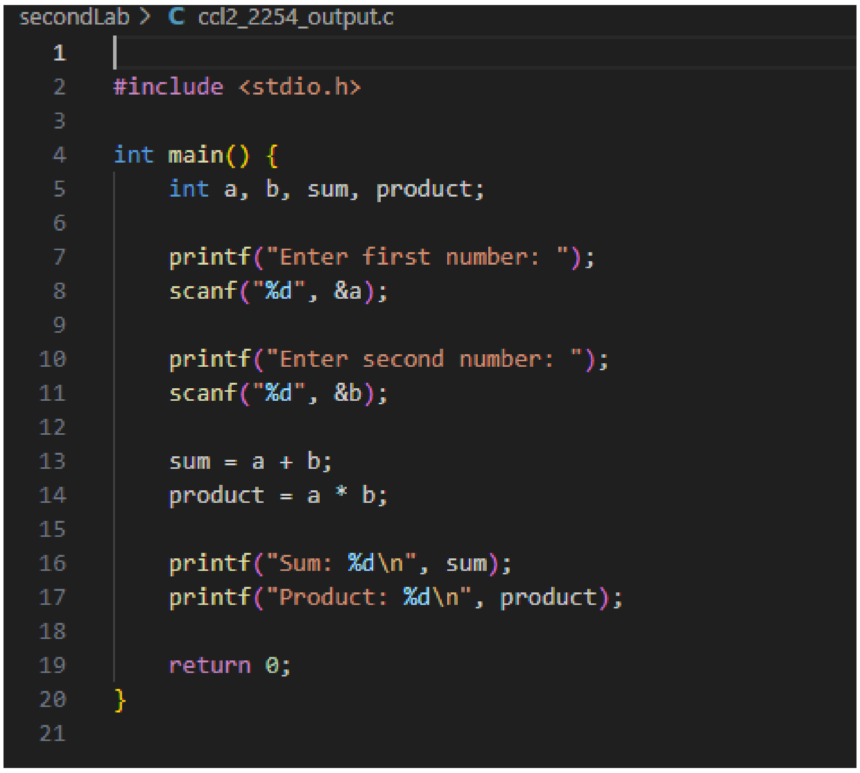


secondLab > ccl2_2254_main.py > ...

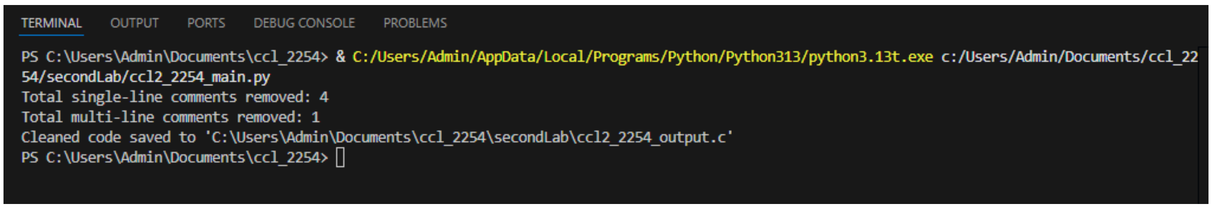
```python
import re

input_file = r"C:\Users\Admin\Documents\ccl_2254\secondLab\ccl_2_2254_input.c"
output_file = r"C:\Users\Admin\Documents\ccl_2254\secondLab\ccl2_2254_output.c"
with open(input_file, 'r') as f:
    code = f.read()

# Count single-line comments (// ...)
single_line_comments = re.findall(r'//.*', code)
single_line_count = len(single_line_comments)

# Count multi-line comments (/* ... */)
multi_line_comments = re.findall(r'/\*.*?\*/', code, flags=re.DOTALL)
multi_line_count = len(multi_line_comments)

# Remove multi-line comments
code_no_multi = re.sub(r'/\*.*?\*/', '', code, flags=re.DOTALL)
# Remove single-line comments
clean_code = re.sub(r'//.*', '', code_no_multi)

with open(output_file, 'w') as f:
    f.write(clean_code)

print(f"Total single-line comments removed: {single_line_count}")
print(f"Total multi-line comments removed: {multi_line_count}")
print(f"Cleaned code saved to '{output_file}'")
```

Figure02: ccl_2_2254_main.py

Output file:



```c

#include <stdio.h>

int main() {
    int a, b, sum, product;

    printf("Enter first number: ");
    scanf("%d", &a);

    printf("Enter second number: ");
    scanf("%d", &b);

    sum = a + b;
    product = a * b;

    printf("Sum: %d\n", sum);
    printf("Product: %d\n", product);

    return 0;
}

```



```
PS C:\Users\Admin\Documents\ccl_2254> & C:/Users/Admin/AppData/Local/Programs/Python/Python313/python3.13t.exe c:/Users/Admin/Documents/ccl_22
54/secondLab/ccl2_2254_main.py
Total single-line comments removed: 4
Total multi-line comments removed: 1
Cleaned code saved to 'C:\Users\Admin\Documents\ccl_2254\secondLab\ccl2_2254_output.c'
PS C:\Users\Admin\Documents\ccl_2254> 
```

Figure03: ccl_2_2254_output.c

Discussion:

In this lab experiment, I wrote a Python program to detect and remove both single-line (//) and multi-line (/*...*/) comments from a C program. I used regular expressions to find out how many single-line and multi-line comments exist in the input C file, and then removed them one by one. First, I read the full C code using file handling, then counted the comments using re.findall() for both types. After counting, I removed the multi-line comments using re.sub() with DOTALL flag so it works over multiple lines, then removed single-line comments the same way. Finally, I wrote the cleaned version of the code to a new output file. The output file has the same logic as the input file, just without any comments. After running the program, it showed how many single-line and multi-line comments were removed, and confirmed that the cleaned code is saved successfully. This task helped me understand how a compiler's preprocessing step works when it removes unnecessary parts like comments before actual compilation starts. I realized that regex is powerful for pattern matching, but it also has some limitations, like it doesn't understand if a comment symbol appears inside a string (e.g., "//" inside quotes). But for this experiment, the code works fine for clean and simple comment structures.