**AIM:**

To design an online simulator that checks whether a given input string (program snippet) is a single/multi-comment or not a comment.

**PROGRAMMING LANGUAGES USED:**

* C Programming Language
* HTML
* CSS
* JavaScript

**THEORY:**

Given a string as user input, representing a program snippet, the task is to check if the given string is a single/multi-line comment or not a comment.

*Types of comments in programs:*

* ***Single Line Comment:***

*Comments preceded by a Double Slash (****‘//’****)*

* ***Multi-line Comment:***

*Comments starting with (****‘/\*’****) and ending with (****‘\*/’****).*

**SIMULATOR LOGIC:**

Check if at the first Index(i.e. index 0) the value is ‘/’ then follow below steps else print “It is not a comment”.

* If line[0] == ‘/’:
  + If line[1] == ‘/’, then print “It is a single line comment”.
  + If line[1] == ‘\*’, then traverse the string and if any adjacent pair of ‘\*’ & ‘/’ is found then print “It is a multi-line comment”.
* Otherwise, print “It is not a comment”.

**INPUT/OUTPUT EXAMPLES:**

* ***Input:****line = “/\* Comment \*/”****Output:****It is a multi-line comment*
* ***Input:****line = “// Comment ”****Output:****It is a single-line comment*

**COMPLEXITY OF SIMULATOR PROGRAM:**

* ***Time Complexity:****O(N)*
* ***Auxiliary Space:****O(1)*

**DISCUSSION:**

We wrote the online simulator using C programming language and integrated it with the website designed using HTML, CSS, JavaScript. The simulator takes a string as user input and checks whether it is a single/multi-line comment or not a comment.

**SOURCES OF ERROR:**

There can be two possible sources of error for this string in the program snippet:

* **Unterminated Multi-Line Comments:**

1. The \*/ (red star followed by slash) indicates the absence of the terminating multi-comment characters.

2. The ~~strikethrough text~~ indicates that compiler will ignore the function (int) and identifier (Comment) during tokenization and instead consider them as part of multi-line comment.

/\* ~~Unterminated Comment \*/ int Comment;~~

It occurs when we do not terminate the comment with a valid set of characters (here the red-coloured \*/). Single line comment does not need to terminate, and it starts by a double slash (//), but, multi-line comment needs to be terminated by \*/ (asterisk and slash). If a multi-line comment is not terminated properly, then during lexical analysis phase of the compiler, the compiler will consider everything after start of the multi-line comment as a comment and ignore them during tokenization (traversing the program from left to right line-by-line and breaking them down into tokens or logical units of a program).

* **Nested Multi-Line Comments:**

1. The Comment \*/ (red coloured text) indicates that the compiler will consider those during lexical analysis phase and count them as 3 tokens.

2. The ~~strikethrough text~~ indicates that compiler will ignore them during tokenization and instead consider them as part of multi-line comment.

~~/\* Comment /\* Nested Comment \*/~~ Comment \*/

In C, multi-line comments, /\* and \*/, do not nest ie, if we want to write a nested comment (comment within comment), the compiler during lexical analysis phase will consider the first /\* as start of multi-line comment and ignore the second /\* syntax as it has still not got an end for the first comment “\*/”. So, the multi-line comment will end after the first \*/ (red coloured). After that, the rest of the comment will not be ignored and will be considered during tokenization until it again gets a start of comment line again.