

## 722. Remove Comments

Difficulty: **Medium**

•Total Accepted:913

•Total Submissions:3.8K

•Contributor:[1337c0d3r](#)

•**Subscribe** to see which companies asked this question.

### Related Topics

String

Given a C++ program, remove comments from it. The program `source` is an array where `source[i]` is the  $i$ -th line of the source code. This represents the result of splitting the original source code string by the newline character `\n`.

In C++, there are two types of comments, line comments, and block comments.

The string `//` denotes a line comment, which represents that it and rest of the characters to the right of it in the same line should be ignored.

The string `/*` denotes a block comment, which represents that all characters until the next (non-overlapping) occurrence of `*/` should be ignored. (Here, occurrences happen in reading order: line by line from left to right.) To be clear, the string `/*` does not yet end the block comment, as the ending would be overlapping the beginning.

The first effective comment takes precedence over others: if the string `//` occurs in a block comment, it is ignored. Similarly, if the string `/*` occurs in a line or block comment, it is also ignored.

If a certain line of code is empty after removing comments, you must not output that line: each string in the answer list will be non-empty.

There will be no control characters, single quote, or double quote characters. For example, `source = "string s = "/* Not a comment. */";"` will not be a test case. (Also, nothing else such as defines or macros will interfere with the comments.)

It is guaranteed that every open block comment will eventually be closed, so `/*` outside of a line or block comment always starts a new comment.

Finally, implicit newline characters can be deleted by block comments. Please see the examples below for details.

After removing the comments from the source code, return the source code in the same format.

### Example 1:

#### Input:

```
source = ["/*Test program */", "int main()", "{ ", " // variable declaration ", "int a, b, c;", "/* This is a test", " multiline ", " comment for ", " testing */", "a = b + c;", "}"]
```

The line by line code is visualized as below:

```
/*Test program */  
  
int main()  
{  
    // variable declaration  
  
    int a, b, c;  
  
    /* This is a test  
       multiline  
       comment for  
       testing */  
  
    a = b + c;  
}
```

**Output:** ["int main()","{ "," ","int a, b, c;","a = b + c;","}"]

The line by line code is visualized as below:

```
int main()  
{  
  
    int a, b, c;  
  
    a = b + c;  
}
```

**Explanation:**

The string `/*` denotes a block comment, including line 1 and lines 6-9. The string `//` denotes line 4 as comments.

**Example 2:**

**Input:**

```
source = ["a/*comment", "line", "more_comment*/b"]
```

**Output:** ["ab"]

**Explanation:** The original source string is "a/\*comment\nline\nmore\_comment\*/b", where we have bolded the newline characters. After deletion, the *implicit* newline characters are deleted, leaving the string "ab", which when delimited by newline characters becomes ["ab"].

**Note:**

- The length of `source` is in the range `[1, 100]`.
- The length of `source[i]` is in the range `[0, 80]`.
- Every open block comment is eventually closed.
- There are no single-quote, double-quote, or control characters in the source code.

```
class Solution {
public:
    vector<string> removeComments(vector<string>& source) {
        vector<string> ans;
        string s;
        bool comment = false;
        for(int i = 0; i < source.size(); i++) {
            for(int j = 0; j < source[i].size(); j++) {
                if(!comment && j + 1 < source[i].size() && source[i][j] ==
                '/' && source[i][j+1]=='/') break;
                else if(!comment && j + 1 < source[i].size() && source[i][j]
                == '/' && source[i][j+1]=='*') comment = true, j++;
                else if(comment && j + 1 < source[i].size() && source[i][j]
                == '*' && source[i][j+1]=='/') comment = false, j++;
                else if(!comment) s.push_back(source[i][j]);
            }

            if(!comment && s.size()) ans.push_back(s), s.clear();
        }
        return ans;
    }
};
```