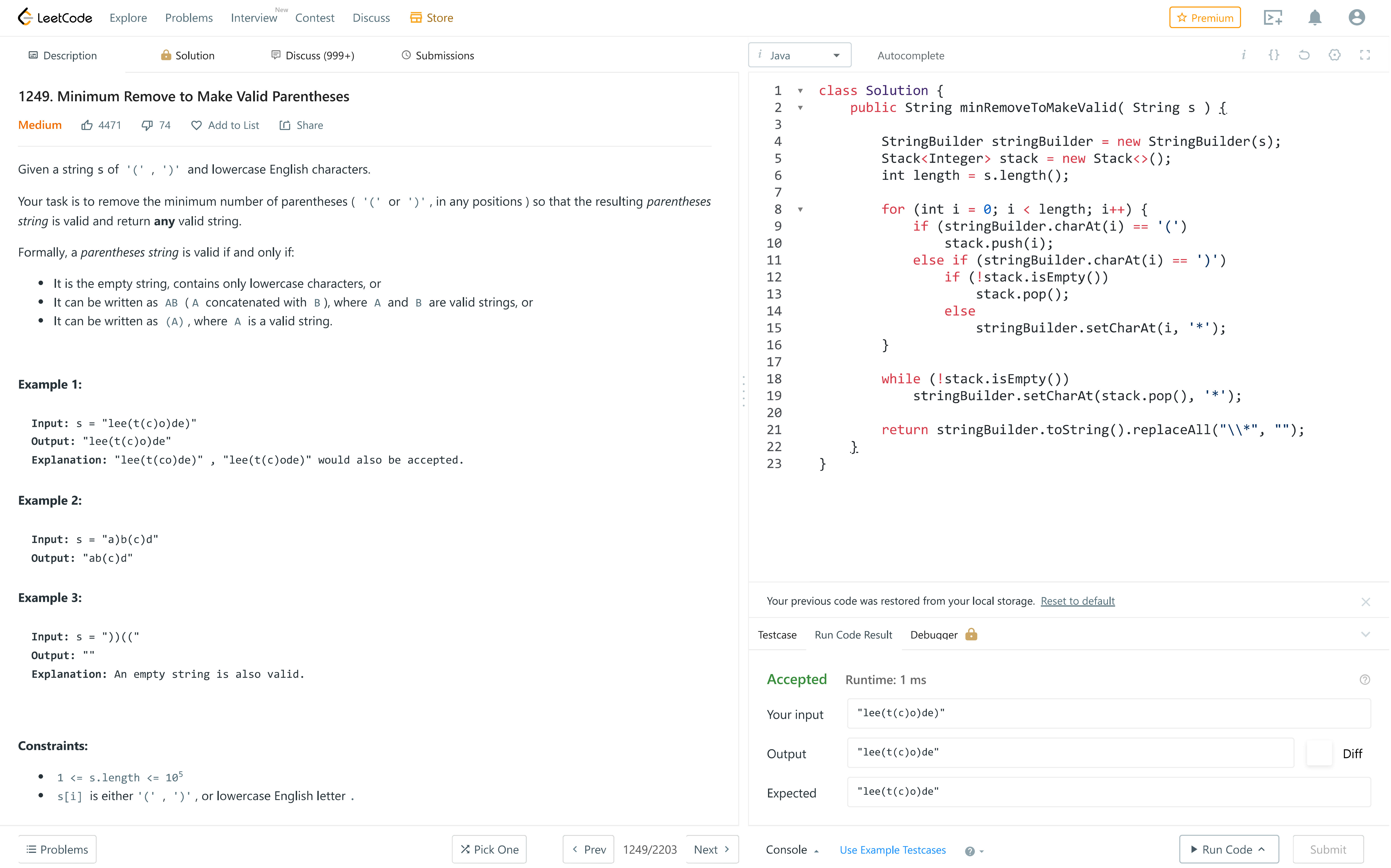

1249. Minimum Remove to Make Valid Parentheses

Given a string s of `'('` , `')'` and lowercase English characters.

Your task is to remove the minimum number of parentheses (`'('` or `')'` , in any positions) so that the resulting *parentheses string* is valid and return **any** valid string.

Formally, a *parentheses string* is valid if and only if:

- It is the empty string, contains only lowercase characters, or
- It can be written as `AB` (`A` concatenated with `B`), where `A` and `B` are valid strings, or
- It can be written as `(A)`, where `A` is a valid string.

Example 1:

```
Input: s = "lee(t(c)o)de)"
Output: "lee(t(c)o)de"
Explanation: "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.
```

Example 2:

```
Input: s = "a)b(c)d"
Output: "ab(c)d"
```

Example 3:

```
Input: s = "))(("
Output: ""
Explanation: An empty string is also valid.
```

Constraints:

- `1 <= s.length <=` 10^5
- `s[i]` is either `'('` , `')'` , or lowercase English letter .

```java
class Solution {
    public String minRemoveToMakeValid( String s ) {

        StringBuilder stringBuilder = new StringBuilder(s);
        Stack<Integer> stack = new Stack<>();
        int length = s.length();

        for (int i = 0; i < length; i++) {
            if (stringBuilder.charAt(i) == '(')
                stack.push(i);
            else if (stringBuilder.charAt(i) == ')')
                if (!stack.isEmpty())
                    stack.pop();
                else
                    stringBuilder.setCharAt(i, '*');
        }

        while (!stack.isEmpty())
            stringBuilder.setCharAt(stack.pop(), '*');

        return stringBuilder.toString().replaceAll("\\*", "");
    }
}
```

Your previous code was restored from your local storage. Reset to default

```
Accepted    Runtime: 1 ms

Your input  "lee(t(c)o)de)"
Output      "lee(t(c)o)de"
Expected    "lee(t(c)o)de"
```