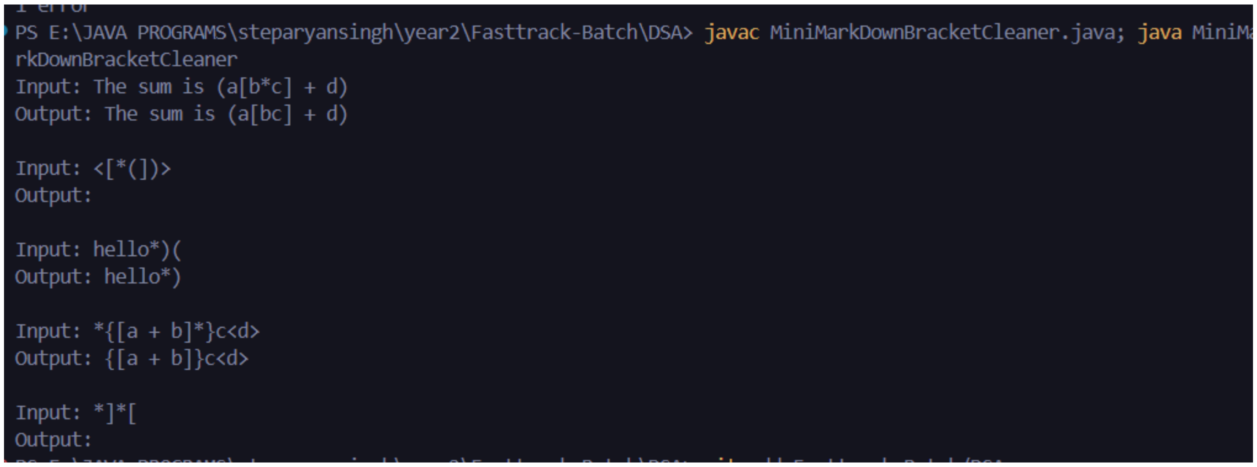
```java
public class MiniMarkDownBracketCleaner {

    String cleanMarkdownBrackets(String s) {
        int n = s.length();
        char[] charArray = s.toCharArray();
        boolean[] keep = new boolean[n];

        Stack<Integer> roundOpen = new Stack<>();
        Stack<Integer> squareOpen = new Stack<>();
        Stack<Integer> curlyOpen = new Stack<>();
        Stack<Integer> angleOpen = new Stack<>();
        Stack<Integer> stars = new Stack<>();

        for (int i = 0; i < n; i++) {
            char c = charArray[i];

            if (c == '(') roundOpen.push(i);
            else if (c == '[') squareOpen.push(i);
            else if (c == '{') curlyOpen.push(i);
            else if (c == '<') angleOpen.push(i);
            else if (c == '*') stars.push(i);

            else if (c == ')') {
                if (!roundOpen.isEmpty()) {
                    keep[roundOpen.pop()] = true;
                    keep[i] = true;
                } else if (!stars.isEmpty()) {
                    keep[stars.pop()] = true;
                    keep[i] = true;
                }
            } else if (c == ']') {
                if (!squareOpen.isEmpty()) {
                    keep[squareOpen.pop()] = true;
                    keep[i] = true;
                } else if (!stars.isEmpty()) {
                    keep[stars.pop()] = true;
                    keep[i] = true;
                }
            } else if (c == '}') {
                if (!curlyOpen.isEmpty()) {
                    keep[curlyOpen.pop()] = true;
                    keep[i] = true;
                } else if (!stars.isEmpty()) {
                    keep[stars.pop()] = true;
                    keep[i] = true;
                }
            } else if (c == '>') {
                if (!angleOpen.isEmpty()) {
                    keep[angleOpen.pop()] = true;
                    keep[i] = true;
                } else if (!stars.isEmpty()) {
                    keep[stars.pop()] = true;
                    keep[i] = true;
                }
            } else {
                keep[i] = true;
            }
        }

        matchRemaining(roundOpen, stars, keep);
        matchRemaining(squareOpen, stars, keep);
        matchRemaining(curlyOpen, stars, keep);
        matchRemaining(angleOpen, stars, keep);

        StringBuilder result = new StringBuilder();
        for (int i = 0; i < n; i++) {
            if (keep[i] && charArray[i] != '*') {
                result.append(charArray[i]);
            }
        }

        if (!isValid(result.toString())) return "";
        return result.toString();
    }

    private void matchRemaining(Stack<Integer> openStack, Stack<Integer>
stars, boolean[] keep) {
        while (!openStack.isEmpty() && !stars.isEmpty()) {
            int openIndex = openStack.pop();
            int starIndex = stars.pop();
            if (openIndex < starIndex) {
                keep[openIndex] = true;
                keep[starIndex] = true;
            } else {
                stars.push(starIndex);
            }
        }
    }

    private boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(' || c == '[' || c == '{' || c == '<') {
                stack.push(c);
            } else if (c == ')') {
                if (stack.isEmpty() || stack.pop() != '(') return false;
            } else if (c == ']') {
                if (stack.isEmpty() || stack.pop() != '[') return false;
            } else if (c == '}') {
                if (stack.isEmpty() || stack.pop() != '{') return false;
            } else if (c == '>') {
                if (stack.isEmpty() || stack.pop() != '<') return false;
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        MiniMarkDownBracketCleaner cleaner = new
MiniMarkDownBracketCleaner();

        System.out.println(cleaner.cleanMarkdownBrackets("The sum is
(a[b*c] + d)"));
        System.out.println(cleaner.cleanMarkdownBrackets("<[*(])>"));
        System.out.println(cleaner.cleanMarkdownBrackets("hello*)("));
    }
}
```



```
PS E:\JAVA PROGRAMS\stepkaryansingh\year2\Fasttrack-Batch\DSA> javac MiniMarkDownBracketCleaner.java; java MiniMa
rkDownBracketCleaner
Input: The sum is (a[b*c] + d)
Output: The sum is (a[bc] + d)

Input: <[*(])>
Output:

Input: hello*)(
Output: hello*)

Input: *{[a + b]*}c<d>
Output: {[a + b]}c<d>

Input: *]*[
Output:
```