Requirements

- Assume the language standard is C# v. 7.0
- We expect good performance.
- If missing requirements details, make reasonable assumptions
- Solution must be simple and compact
  - No defensive coding, no comments, no unrequested features.
  - Only one file 10-20 lines of code
- Work only inside Google Docs: no external editor/IDE/debugger, no copy-paste to/from such an editor. We must see the flow of how you write the code.

Note: you have a total of 30 minutes for both questions!
Ensure you have filled all pages.

Task 1

Implement function

int verify(string text)

which verifies whether parentheses within text are
correctly nested. You need to consider three kinds: (), [], <> and only these kinds.

Examples

```
verify("---(++++)----") -> 1
verify("") -> 1
verify("before ( middle []) after ") -> 1
verify(") (") -> 0
verify("<(   >)") -> 0
verify("(  [  <>  ()  ]  <>  )") -> 1
verify("   (      [)") -> 0
```

Answer

```
int verify(string text) {
        var stack = new Stack();

        for (int i = 0; i < text.Length; i++) {
                var elem = text[i];
                if (elem == ‘(’ || elem == ‘[’ || elem == ‘<’) {
                        stack.Push(elem);
```

```
            } else if (elem == ‘)’ || elem == ‘]’ || elem == ‘>’) {
                var peek = (stack.Count > 0) ? stack.Peek().ToString()[0] : char.MinValue;

                If ((peek == ‘(’ && elem == ‘)’) || (peek == ‘[’ && elem == ‘]’) || (peek == ‘<’
        && elem == ‘>’)) {
                stack.Pop();
            } else {
                return 0;
            }
            }
        }

        return (stack.Count == 0 ? 1 : 0);
    }
```

Task 2

Simplify the implementation below as much as you can.
Even better if you can also improve performance as part of the simplification!
FYI: This code is over 35 lines and over 300 tokens, but it can be written in 5 lines and in less than 60 tokens.

Code

```
public static int func(String s, String a, String b){
    Regex rx = new Regex(@"^$");
    MatchCollection matches = rx.Matches(s);
    if (matches.Count > 0)
        return -1;
    else
    {
        int i = s.Length - 1;
        int aIndex =- 1;
        int bIndex =- 1;
        while ((aIndex == -1) && (bIndex == -1) && (i >= 0))
        {
            if (s.Substring(i, Math.Max(Math.Min(i+1, s.Length-i)-i, 1)).Equals(a))
                aIndex = i;
            if (s.Substring(i, Math.Max(Math.Min(i+1, s.Length-i)-i, 1)).Equals(b))
                bIndex = i;
            i--;
        }
        if (aIndex != -1)
        {
            if (bIndex == -1)
                return aIndex;
            else
                return Math.Max(aIndex, bIndex);
        }
        else
        {
            if (bIndex != -1)
                return bIndex;
            else
                return -1;
        }
    }
}
```

Answer

```
public static int func(String s, String a, String b) {
        var lastIndexA = s.LastIndexOf(a);
        var lastIndexB = s.LastIndexOf(b);
        if (lastIndexA != -1 && lastIndexB != -1) return Math.Max(lastIndexA, lastIndexB);
        if (lastIndexA != -1) return lastIndexA;
        return lastIndexB;
}
```