题目描述：

给定一个小写字母组成的字符串 s，请找出字符串中两个不同位置的字符作为分割点，使得字符串分成的三个连续子串且子串权重相等，注意子串不包含分割点。
若能找到满足条件的两个分割点，请输出这两个分割点在字符串中的位置下标，若不能找到满足条件的分割点请返回 0,0。

子串权重计算方式为：子串所有字符的 ASCII 码数值之和。

输入描述：

输入为一个字符串，字符串由 a~z，26 个小写字符组成，5 <= 字符串长度 <= 200。

输出描述：

输出为两个分割点在字符串中的位置下标，以逗号分隔

补充说明：
只考虑唯一解，不存在一个输入多种输出解的情况

示例 1

输入：

```
acdbbbca
```

输出：

```
2,5
```

说明：

以位置 2 和 5 作为分割点，将字符串分割为 ac，bb，ca 三个子串，每一个的子串权重都为 196，输出为：2,5

示例 2

输入：

```
abcabc
```

输出：

```
0,0
```

说明：

找不到符合条件的分割点，输出为0,0

```
package main

import "fmt"

func main() {
	s := ""
	fmt.Scanln(&s)
	d1, d2 := Cal(s)
	fmt.Printf("%d,%d\n", d1, d2)
}

func Cal(s string) (int, int) {
	length := len(s)
	for i := 0; i < length; i++ {
		for j := i + 2; j < length-1; j++ {
			sum1 := Sum(s[0:i])
			sum2 := Sum(s[i+1 : j])
			sum3 := Sum(s[j+1:])
			if sum1 == sum2 && sum2 == sum3 {
				return i, j
			}
		}
	}
	return 0, 0
}

func Sum(s string) int {
	var result int
	for _, e := range s {
		result += int(e)
	}
	return result
}
```