Question **1**

Correct

Marked out of 10.00

Given a string S as input which consists only of digits from 0 to 9, print the longest substring such that the sum of the digits in the first half and the second half is the same. Print -1 if such a substring does not exist.

Input Format:

The first line contains S

Output Format:

The first line contains the longest substring as per the rules defined above or -1.

Boundary Conditions:

1 <= Length of S <= 100

Example Input/Output 1:

Input:

123123

Output:

123123

Explanation:

The first half is 123 and the second half is 123. Hence the sum of the digits is equal.

Example Input/Output 2:

Input:

1538024

Output:

5380

Explanation:

The first half is 53 and the second half is 80. The sum of the digits is 8 in both the halves.

Example Input/Output 3:

Input:

12345

Output:

-1

Example Input/Output 4:

Input:

989898989

Output:

98989898

Explanation:

Here both 98989898 and 89898989 are of same length. But due to order of occurrence 98989898 is printed as the output.

For example:

Input	Result
123123	123123
1538024	5380
12345	-1
989898989	98989898

Answer: (penalty regime: 0 %)

```
def find(s):
    n=len(s)
    max_len=0
    result="-1"

    for i in range(2,n+1,2):
        for j in range(n-i+1):
            mid=j+i//2
            first=s[j:mid]
            second=s[mid:j+i]


            if sum(map(int,first)) == sum(map(int,second)):
                if i>max_len:
                    max_len=i
                    result=s[j:j+i]
    print(result)

s=input().strip()
find(s)
```

	Input	Expected	Got	
✓	123123	123123	123123	✓
✓	1538024	5380	5380	✓
✓	12345	-1	-1	✓
✓	989898989	98989898	98989898	✓

Passed all tests! ✓