Problem Statement

Given a string of digits, find the minimum number of additions required for the string to equal some target number. Each addition is the equivalent of inserting a plus sign somewhere into the string of digits. After all plus signs are inserted, evaluate the sum as usual. For example, consider the string "12" (quotes for clarity). With zero additions, we can achieve the number 12. If we insert one plus sign into the string, we get "1+2", which evaluates to 3. So, in that case, given "12", a minimum of 1 addition is required to get the number 3. As another example, consider "303" and a target sum of 6. The best strategy is not "3+0+3", but "3+03". You can do this because leading zeros do not change the result.

Write a class QuickSums that contains the method minSums, which takes a String **numbers** and an int **sum**. The method should calculate and return the minimum number of additions required to create an expression from **numbers** that evaluates to **sum**. If this is impossible, return -1.

Definition

Class:	QuickSums
Method:	minSums
Parameters:	String, int
Returns:	int
Method signature:	int minSums(String numbers, int sum)
(be sure your method is public)	

Constraints

- **numbers** will contain between 1 and 10 characters, inclusive.
- Each character in **numbers** will be a digit.
- **sum** will be between 0 and 100, inclusive.

Examples

0)

```
"99999"
45
Returns: 4
```

In this case, the only way to achieve 45 is to add 9+9+9+9+9. This requires 4 additions.

1)

```
"1110"
3
Returns: 3
```

Be careful with zeros. 1+1+1+0=3 and requires 3 additions.

2)

```
"0123456789"
45
Returns: 8
```

3)

```
"99999"
```

```
100
```
```
Returns: -1
```

4)

```
"382834"
100
```
```
Returns: 2
```

There are 3 ways to get 100. They are 38+28+34, 3+8+2+83+4 and 3+82+8+3+4. The minimum required is 2.

5)

```
"9230560001"
71
```
```
Returns: 4
```