Alice has a binary string. She thinks a binary string is beautiful if and only if it doesn't contain the substring `"010"`.

In one step, Alice can change a 0 to a 1 or vice versa. Count and print the minimum number of steps needed to make Alice see the string as beautiful.

For example, if Alice's string is $b = 010$ she can change any one element and have a beautiful string.

Function Description

Complete the *beautifulBinaryString* function in the editor below. It should return an integer representing the minimum moves required.

beautifulBinaryString has the following parameter(s):

- b: a string of binary digits

Input Format

The first line contains an integer n, the length of binary string.
The second line contains a single binary string b.

Constraints

- $1 \leq n \leq 100$
- $b[i] \in \{0, 1\}$.

Output Format

Print the minimum number of steps needed to make the string beautiful.

Sample Input 0

```
7
0101010
```

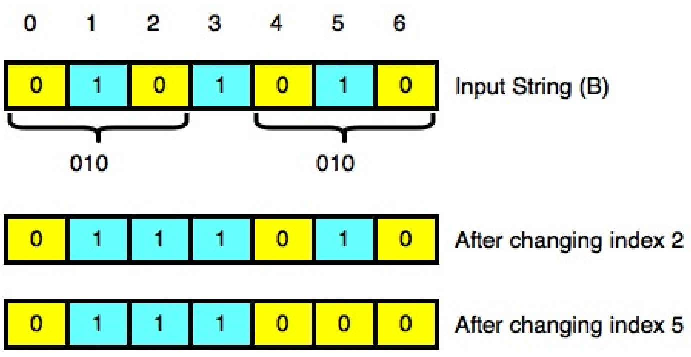
Sample Output 0

```
2
```

Explanation 0:

In this sample, $b = "0101010"$

The figure below shows a way to get rid of each instance of $"010"$:



Because we were able to make the string beautiful by changing 2 characters ($b[2]$ and $b[5]$), we print 2.

Sample Input 1

```
5
01100
```

Sample Output 1

```
0
```

Sample Case 1:

In this sample $b = "01100"$

Explanation 1

The substring $"010"$ does not occur in b, so the string is already beautiful and we print 0.

Sample Input 2

```
10
0100101010
```

Sample Output 2

```
3
```

Explanation 2

In this sample $b = "0100101010"$

One solution is to change the values of $b[2], b[5] \text{ and } b[9]$ to form a beautiful string.