A. Stones on the Table

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

There are *n* stones on the table in a row, each of them can be red, green or blue. Count the minimum number of stones to take from the table so that any two neighboring stones had different colors. Stones in a row are considered neighboring if there are no other stones between them.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 50) — the number of stones on the table.

The next line contains string *s*, which represents the colors of the stones. We'll consider the stones in the row numbered from 1 to *n* from left to right. Then the *i*-th character *s* equals "R", if the *i*-th stone is red, "G", if it's green and "B", if it's blue.

**Output**

Print a single integer — the answer to the problem.

**Examples**

**input**

3  
RRG

**output**

1

**input**

5  
RRRRR

**output**

4

**input**

4  
BRBG

**output**

0

<http://codeforces.com/problemset/problem/266/A>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int n = int.Parse(Console.ReadLine());

string s = Console.ReadLine();

// int n = 10;// int.Parse(Console.ReadLine());

// string s = "RRRRRRRRRR"; // Console.ReadLine();

//int n = 3;

//string s = "RRG";

//List<char> lista = new List<char>();

//for (int i = 0; i < s.Length; i++)

//{

// lista.Add(s[i]);

//}

int i = 0;

string concat = "";

while (i < s.Length)

{

while (i + 1 < s.Length && s[i] == s[i + 1])

{

i++;

}

concat += s[i];

i++;

}

Console.WriteLine(s.Length - concat.Length);

Console.ReadLine();

}

}

}