CodeWars Finals

QUESTION # 1:

Cookie Clicker Alpha

Cookie Clicker is a Javascript game by Orteil, where players click on a picture of a giant cookie. Clicking on the giant cookie gives them cookies. They can spend those cookies to buy buildings. Those buildings help them get even more cookies.

In this problem, you start with 0 cookies. You gain cookies at a rate of 2 cookies per second, by clicking on a giant cookie. Any time you have at least **C** cookies, you can buy a cookie farm. Every time you buy a cookie farm, it costs you **C** cookies and gives you an extra **F** cookies per second.

Once you have **X** cookies that you haven't spent on farms, you win! Figure out how long it will take you to win if you use the best possible strategy.

Example

Suppose **C**=500.0, **F**=4.0 and **X**=2000.0. Here's how the best possible strategy plays out:

1. You start with 0 cookies, but producing 2 cookies per second.
2. After **250** seconds, you will have **C**=500 cookies and can buy a farm that produces **F**=4 cookies per second.
3. After buying the farm, you have 0 cookies, and your total cookie production is 6 cookies per second.
4. The next farm will cost 500 cookies, which you can buy after about **83.3333333**seconds.
5. After buying your second farm, you have 0 cookies, and your total cookie production is 10 cookies per second.
6. Another farm will cost 500 cookies, which you can buy after **50** seconds.
7. After buying your third farm, you have 0 cookies, and your total cookie production is 14 cookies per second.
8. Another farm would cost 500 cookies, but it actually makes sense not to buy it: instead you can just wait until you have **X**=2000 cookies, which takes about **142.8571429** seconds.

Total time: 250 + 83.3333333 + 50 + 142.8571429 = 526.1904762 seconds.

Notice that you get cookies continuously: so, 0.1 seconds after the game starts you'll have 0.2 cookies, and π seconds after the game starts you'll have 2π cookies.

Problem setters: Amir Qadir & Amar Kumar

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line contains three space-separated real-valued numbers: **C**, **F** and **X**, whose meanings are described earlier in the problem statement.

C, **F** and **X** will each consist of at least 1 digit followed by 1 decimal point followed by from 1 to 5 digits. There will be no leading zeroes.

Output

For each test case, output one line containing "Case #x: y", where x is the test case number (starting from 1) and y is the minimum number of seconds it takes before you can have **X** delicious cookies.

We recommend outputting y to 7 decimal places.

Constraints:

$1 \leq \mathbf{T} \leq 100$.

$1 \leq \mathbf{C} \leq 500$.
$1 \leq \mathbf{F} \leq 4$.
$1 \leq \mathbf{X} \leq 2000$.

Sample Input

```
4
30.0 1.0 2.0
30.0 2.0 100.0
30.50000 3.14159 1999.19990
500.0 4.0 2000.0
```

Sample Output

```
Case #1: 1.0000000
Case #2: 39.1666667
Case #3: 63.9680013
Case #4: 526.1904762
```

Problem setters: Amir Qadir & Amar Kumar

QUESTION # 2:

QUEUE AT THE SCHOOL

During the break the schoolchildren, boys and girls, formed a queue of n people in the canteen. Initially the children stood in the order they entered the canteen. However, after a while the boys started feeling awkward for standing in front of the girls in the queue and they started letting the girls move forward each second.

Let's describe the process more precisely. Let's say that the positions in the queue are sequentially numbered by integers from 1 to n, at that the person in the position number 1 is served first. Then, if at time x a boy stands on the i-th position and a girl stands on the (i + 1)-th position, then at time x + 1 the i-th position will have a girl and the (i + 1)-th position will have a boy. The time is given in seconds.

You've got the initial position of the children, at the initial moment of time. Determine the way the queue is going to look after t seconds.

Input

The first line contains two integers n and t ($1 \le n, t \le 50$), which represent the number of children in the queue and the time after which the queue will transform into the arrangement you need to find.

The next line contains string s, which represents the schoolchildren's initial arrangement. If the i-th position in the queue contains a boy, then the i-th character of string s equals "B", otherwise the i-th character equals "G".

Output

Print string a, which describes the arrangement after t seconds. If the i-th position has a boy after the needed time, then the i-th character a must equal "B", otherwise it must equal "G".

Sample Input

```
5 1
BGGBG
```

Problem setters: Amir Qadir & Amar Kumar

Sample Output

```
GBGGB
```

QUESTION # 3:

FAIR AND SQUARE

Little John likes palindromes, and thinks them to be fair (which is a fancy word for nice). A *palindrome* is just an integer that reads the same backwards and forwards - so 6, 11 and 121 are all palindromes, while 10, 12, 223 and 2244 are not (even though 010=10, we don't consider leading zeroes when determining whether a number is a palindrome).

He recently became interested in squares as well, and formed the definition of a *fair and square* number - it is a number that is a palindrome **and** the *square of a palindrome* at the same time. For instance, 1, 9 and 121 are fair and square (being palindromes and squares, respectively, of 1, 3 and 11), while 16, 22 and 676 are **not** fair and square: 16 is not a palindrome, 22 is not a square, and while 676 is a palindrome and a square number, it is the square of 26, which is not a palindrome.

Now he wants to search for bigger fair and square numbers. Your task is, given an interval Little John is searching through, to tell him how many fair and square numbers are there in the interval, so he knows when he has found them all.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow. Each line contains two integers, **A** and **B** - the endpoints of the interval Little John is looking at.

Output

For each test case, output one line containing "Case #x: y", where x is the case number (starting from 1) and y is the number of fair and square numbers greater than or equal to **A** and smaller than or equal to **B**.

Constraints

1 ≤ **T** ≤ 100.
1 ≤ **A** ≤ **B** ≤ 1000.

Sample Input

```
3
1 4
10 120
100 1000
```

Problem setters: Amir Qadir & Amar Kumar

Sample Output

```
Case #1: 2
Case #2: 0
Case #3: 2
```

Problem setters: Amir Qadir & Amar Kumar

QUESTION # 4:

Fair Rations

You are the benevolent ruler of Rankhacker Castle, and today you're distributing bread to a straight line of N subjects. Each i^{th} subject (where $1 \le i \le N$) already has B_i loaves of bread.

Times are hard and your castle's food stocks are dwindling, so you must distribute as few loaves as possible according to the following rules:

1. Every time you give a loaf of bread to some person i, you must also give a loaf of bread to the person immediately in front of or behind them in the line (i.e., persons $i+1$ or $i-1$). In other words, you can only give a loaf of bread each to two adjacent people at a time.
2. After all the bread is distributed, each person must have an even number of loaves.

Given the number of loaves already held by each citizen, find and print the minimum number of loaves you must distribute to satisfy the two rules above. If this is not possible, print NO.

Input Format

The first line contains an integer ***N***, denoting the number of subjects in the bread line. The second line contains ***N*** space-separated integers describing the respective loaves of bread already possessed by each person in the line (i.e., $B_1, B_2, \ldots, B_N$).

Constraints

- $2 \le N \le 1000$
- $1 \le B_i \le 10$, where $1 \le i \le N$

Output Format

Print a single integer denoting the minimum number of loaves you must distribute to adjacent people in the line so that every person has an even number of loaves; if it's not possible to do this, print NO.

Sample Input 0

```
5
2 3 4 5 6
```

Sample Output 0

```
4
```

Sample Input 1

```
2
1 2
```

Sample Output 1

```
NO
```

Explanation

Sample Case 0:

The initial distribution is $(2, 3, 4, 5, 6)$. You can satisfy the problem's requirements by performing the following actions:

1. Give 1 loaf of bread each to the second and third people so that the distribution becomes **(2,4,5,5,6)**
2. Give 1 loaf of bread each to the third and fourth people so that the distribution becomes **(2,4,6,6,6)**

Because each of the N subjects now has an even number of loaves, we can stop distributing bread. We then print the total number of loaves distributed, which is 4, on a new line.

Sample Case 1:

The initial distribution is $(1, 2)$. As there are only 2 people in the line, any time you give one person a loaf you must always give the other person a loaf. Because the first person has an odd number of loaves and the second person has an even number of loaves, no amount of distributed loaves will ever result in both subjects having an even number of loaves. Thus, we print NO as our answer.

QUESTION #5:

Find Digits

An integer **d**, is a divisor of an integer **n** if the remainder of **n/d = 0**.
Given an integer, for each digit that makes up the integer determine whether it is a divisor. Count the number of divisors occurring within the integer.
Note: Each digit is considered to be unique, so each occurrence of the same digit should be counted (e.g. for n = 111, 1 is a divisor of 111 each time it occurs so the answer is 3).

Input Format

The first line is an integer **t**, indicating the number of test cases.

The t subsequent lines each contain an integer **n**.

Constraints

$1 <= t <= 15$

$0 < n < 10^9$

Output Format

For every test case, count the number of digits in n that are divisors of n. Print each answer on a new line.

Sample Input

```
2
12
1012
```

Sample Output

```
2
3
```

Problem setters: Amir Qadir & Amar Kumar

Explanation

The number 12 is broken into two digits, 1 and 2. When 12 is divided by either of those two digits, the remainder is 0 so they are both divisors.

The number 1012 is broken into four digits, 1, 0, 1, and 2. 1012 is evenly divisible by its digits 1, 1, and 2, but it is not divisible by 0 as division by zero is undefined.