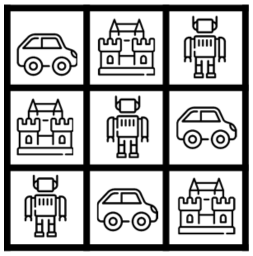
A. Latin Squares



A Latin Square is an n-by-n array filled with n different digits, each digit occurring exactly once in each row and once in each column. (The name "Latin Square" was inspired by the work of Leonhard Euler, who used Latin characters in his papers on the topic.)

A Latin Square is said to be in *reduced form* if both its top row and leftmost column are in their natural order. The natural order of a set of digits is by increasing value.

Your team is to write a program that will read an n-by-n array, and determine whether it is a Latin Square, and if so, whether it is in reduced form.

Input

The first line of input contains a single integer n ($2 \leq n \leq 36$). Each of the next n lines contains n digits in base n, with the normal digits '`0`' through '`9`' for digit values below 10 and uppercase letters '`A`' through '`Z`' representing digit values 10 through 35. All digits will be legal for base n; for instance, if n is 3, the only legal characters in the n input lines describing the square will be '`0`', '`1`', and '`2`'.

Output

If the given array is not a Latin Square, print "`No`" on a single line (without quotation marks). If it is a Latin Square, but not in reduced form, print "`Not Reduced`" on a single line (without quotation marks). If it is a Latin Square in reduced form, print "`Reduced`" on a single line (without quotation marks).

Note: The input and output descriptions above are not complete. Some of the test cases contain multiple sequential instances of the type described above. In this case the output will be the answer to each of those test cases sequentially. For details see the extra example on the next page.

Sample Input and Output

```
3
012
120
201
```

```
Reduced
```

```
4
3210
0123
2301
1032
```

```
Not Reduced
```

```
11
0123458372A
A9287346283
0285475A834
84738299A02
1947584037A
65848430002
038955873A8
947530200A8
93484721084
95539A92828
04553883568
```

```
No
```

```
4
3210
0123
2301
1032
2
11
10
```

```
Not Reduced
No
```

B. Unloaded Die

Consider a six-sided die, with sides labeled 1 through 6. We say the die is *fair* if each of its sides is equally likely to be face up after a roll. We say the die is *loaded* if it isn't fair. For example, if the side marked 6 is twice as likely to come up as than any other side, we are dealing with a loaded die.

For any die, define the *expected result* of rolling the die to be equal to the average of the values of the sides, weighted by the probability of those sides coming up. For example, all six sides of a fair die are equally likely to come up, and thus the expected result of rolling it is $(1+2+3+4+5+6)/6 = 3.5$.

You are given a loaded die, and you would like to *unload* it to make it more closely resemble a fair die. To do so, you can erase the number on one of the sides, and replace it with a new number which does not need to be an integer or even positive. You want to do so in such a way that

- The expected result of rolling the die is 3.5, just like a fair die.
- The difference between the old label and the new label on the side you change is as small as possible.

Input

The input consists of a single line containing six space-separated nonnegative real numbers $v_1 \dots v_6$, where v_i represents the probability that side i (currently labeled by the number i) is rolled.

It is guaranteed that the given numbers will sum to 1.

Output

Print, on a single line, the absolute value of the difference between the new label and old label, rounded and displayed to exactly three decimal places.

Sample Input and Output

`0.16666 0.16667 0.16667 0.16666 0.16667 0.16667`	`0.000`

`0.2 0.2 0.1 0.2 0.2 0.1`	`1.000`

C. Halfway

A friend of yours has written a program that compares every pair of a list of items. With n items, it works as follows. First, it prints a 1, and it compares item 1 to items $2, 3, 4, \ldots, n$. It then prints a 2, and compares item 2 to items $3, 4, 5, \ldots, n$. It continues like that until every pair of items has been compared exactly once. If it compares item x to item y, it will not later compare item y to item x. Also, it does not compare any item to itself.

Your friend wants to know when his program is *halfway done*. For a program that makes an odd number of total comparisons, this is when it is doing the middle comparison. For a program that makes an even number of total comparisons, this is when it is doing the first of the two middle comparisons.

What will the last number printed be when the program is halfway done?

Note that since the earlier items have more comparisons than the later items, the answer is not simply $n/2$.

Input

The input consists of a single line containing the integer n ($2 \leq n \leq 10^9$).

Output

Print, on a single line, the last number your friend's program prints when it is halfway done.

Sample Input and Output

`4`	`1`

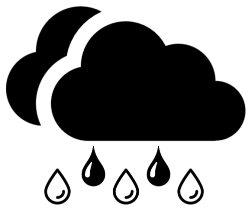
7	2

10	3

1919	562

290976843	85225144

D. Purple Rain



Purple rain falls in the magic kingdom of Linearland which is a straight, thin peninsula.

On close observation however, Professor Nelson Rogers finds that the purple rain is actually a mix of red and blue raindrops.

In his zeal, he records the location and color of the raindrops in different locations along the peninsula. Looking at the data, Professor Rogers wants to know which part of Linearland had the "least" purple rain.

After some thought, he decides to model this problem as follows. Divide the peninsula into n sections and number them west to east from 1 to n. Then, describe the raindrops as a sequence of `R` and `B`, depending on whether the rainfall in each section is primarily red or blue. Finally, find a subsequence of contiguous sections where the difference between the number of `R` and the number of `B` is maximized.

Input

The input consists of a single line containing a string of n characters ($1 \leq n \leq 10^5$), describing the color of the raindrops in sections 1 to n.

It is guaranteed that the string consists of uppercase ASCII letters '`R`' and '`B`' only.

Output

Print, on a single line, two space-separated integers that describe the starting and ending positions of the part of Linearland that had the least purple rain. These two numbers should describe an inclusive range; both numbers you print describe sections included in the range.

If there are multiple possible answers, print the one that has the westernmost starting section. If there are multiple answers with the same westernmost starting section, print the one with the westernmost ending section.

Sample Input and Output

BBRRBRRBRB	3 7

BBRBBRRB	1 5

E. Crusher's Code

Wesley Crusher is the teaching assistant for Introduction to Algorithms. During his first class, the cadets were asked to come up with their own sorting algorithms. Monty came up with the following code:

```
while (!sorted(a)) {
   int i = random(n) ;
   int j = random(n) ;
   if (a[min(i,j)] > a[max(i,j)])
      swap(a[i], a[j]) ;
}
```

Carlos, inspired, came up with the following code:

```
while (!sorted(a)) {
   int i = random(n-1) ;
   int j = i + 1 ;
   if (a[i] > a[j])
      swap(a[i], a[j]) ;
}
```

Wesley needs to determine which algorithm is better.

For a given input array of up to 8 values, calculate and print the expected number of iterations for each algorithm. That is, on average, how many iterations should each algorithm take for the given input?

Input

The first line contains T, the number of test cases: $2 \leq T \leq 100$.

Each test case is given on a single line. The first value is N, the number of array elements; $2 \leq N \leq 8$. This is followed on the same line by N integer array elements. The array elements will have values between 0 and 100 inclusive. The array elements may not be distinct.

Output

For each test case, print out the expected number of iterations for Monty's algorithm and for Carlos's algorithm, as shown in the sample output section. There should be exactly one space between words and no spaces at the start of each line or at the end of each line. There should be exactly six digits after the decimal point. Rounding should be to nearest representable value.

Sample Input

```
12
2 1 2
2 2 1
3 1 2 3
3 3 2 1
4 1 2 3 4
4 4 3 2 1
4 2 1 4 3
5 1 1 1 1 1
5 5 4 3 2 1
8 8 7 6 5 4 3 2 1
8 3 1 4 1 5 9 2 6
8 2 7 1 8 2 8 1 8
```

Sample Output

```
Monty 0.000000 Carlos 0.000000
Monty 2.000000 Carlos 1.000000
Monty 0.000000 Carlos 0.000000
Monty 6.000000 Carlos 5.000000
Monty 0.000000 Carlos 0.000000
Monty 14.666667 Carlos 12.500000
Monty 12.000000 Carlos 4.500000
Monty 0.000000 Carlos 0.000000
Monty 26.382275 Carlos 23.641975
Monty 89.576273 Carlos 79.496510
Monty 79.161905 Carlos 33.422840
Monty 63.815873 Carlos 38.910494
```

F. Bones's Battery

Bones is investigating what electric shuttle is appropriate for his mom's school district vehicle. Each school has a charging station. It is important that a trip from one school to any other be completed with no more than K rechargings. The car is initially at zero battery and must always be recharged at the start of each trip; this counts as one of the K rechargings. There is at most one road between each pair of schools, and there is at least one path of roads connecting each pair of schools. Given the layout of these roads and K, compute the necessary range required of the electric shuttle.

Input

Input begins with a line with one integer T ($1 \leq T \leq 50$) denoting the number of test cases. Each test case begins with a line containing three integers N, K, and M ($2 \leq N \leq 100$, $1 \leq K \leq 100$), where N denotes the number of schools, K denotes the maximum number of rechargings permitted per trip, and M denotes the number of roads. Next follow M lines each with three integers u_i, v_i, and d_i ($0 \leq u_i, v_i < N$, $u_i \neq v_i$, $1 \leq d_i \leq 10^9$) indicating that road i connects schools u_i and v_i (0-indexed) bidirectionally with distance d_i.

Output

For each test case, output one line containing the minimum range required.

Sample Input

```
2
4 2 4
0 1 100
1 2 200
2 3 300
3 0 400
10 2 15
0 1 113
1 2 314
2 3 271
3 4 141
4 0 173
5 7 235
7 9 979
9 6 402
6 8 431
8 5 462
0 5 411
1 6 855
2 7 921
3 8 355
4 9 113
```

Sample Output

```
300
688
```