CSEC-ASTU Group-Contest Round-I

A. GCD Arrays

2 seconds, 256 megabytes

Consider the array a composed of all the integers in the range $[l, r]$. For example, if $l = 3$ and $r = 7$, then $a = [3, 4, 5, 6, 7]$.

Given l, r, and k, is it possible for $\gcd(a)$ to be greater than 1 after doing the following operation at most k times?

- Choose 2 numbers from a.
- Permanently remove one occurrence of each of them from the array.
- Insert their product back into a.

$\gcd(b)$ denotes the greatest common divisor (GCD) of the integers in b.

Input

The first line of the input contains a single integer t $(1 \le t \le 10^5)$ — the number of test cases. The description of test cases follows.

The input for each test case consists of a single line containing 3 non-negative integers l, r, and k $(1 \le l \le r \le 10^9,\ \ 0 \le k \le r - l)$.

Output

For each test case, print "YES" if it is possible to have the GCD of the corresponding array greater than 1 by performing at most k operations, and "NO" otherwise (case insensitive).

input
```
9
1 1 0
3 5 1
13 13 0
4 4 0
3 7 4
4 10 3
2 4 0
1 7 3
1 5 3
```
output
```
NO
NO
YES
YES
YES
YES
NO
NO
YES
```

For the first test case, $a = [1]$, so the answer is "NO", since the only element in the array is 1.

For the second test case the array is $a = [3, 4, 5]$ and we have 1 operation. After the first operation the array can change to: $[3, 20]$, $[4, 15]$ or $[5, 12]$ all of which having their greatest common divisor equal to 1 so the answer is "NO".

For the third test case, $a = [13]$, so the answer is "YES", since the only element in the array is 13.

For the fourth test case, $a = [4]$, so the answer is "YES", since the only element in the array is 4.

B. Minimum Difficulty

2 seconds, 256 megabytes

Mike is trying rock climbing but he is awful at it.

There are n holds on the wall, i-th hold is at height a_i off the ground. Besides, let the sequence a_i increase, that is, $a_i < a_{i+1}$ for all i from 1 to $n - 1$; we will call such sequence a track. Mike thinks that the track a_1, ..., a_n has difficulty $d = \max_{1 \le i \le n-1}(a_{i+1} - a_i)$. In other words, difficulty equals the maximum distance between two holds that are adjacent in height.

Today Mike decided to cover the track with holds hanging on heights a_1, ..., a_n. To make the problem harder, Mike decided to remove one hold, that is, remove one element of the sequence (for example, if we take the sequence $(1, 2, 3, 4, 5)$ and remove the third element from it, we obtain the sequence $(1, 2, 4, 5)$). However, as Mike is awful at climbing, he wants the final difficulty (i.e. the maximum difference of heights between adjacent holds after removing the hold) to be as small as possible among all possible options of removing a hold. The first and last holds **must** stay at their positions.

Help Mike determine the minimum difficulty of the track after removing one hold.

Input

The first line contains a single integer n $(3 \le n \le 100)$ — the number of holds.

The next line contains n space-separated integers a_i $(1 \le a_i \le 1000)$, where a_i is the height where the hold number i hangs. The sequence a_i is increasing (i.e. each element except for the first one is strictly larger than the previous one).

Output

Print a single number — the minimum difficulty of the track after removing a single hold.

input
```
3
1 4 6
```
output
```
5
```

input
```
5
1 2 3 4 5
```
output
```
2
```

input
```
5
1 2 3 7 8
```
output
```
4
```

In the first sample you can remove only the second hold, then the sequence looks like $(1, 6)$, the maximum difference of the neighboring elements equals 5.

In the second test after removing every hold the difficulty equals 2.

In the third test you can obtain sequences $(1, 3, 7, 8)$, $(1, 2, 7, 8)$, $(1, 2, 3, 8)$, for which the difficulty is 4, 5 and 5, respectively. Thus, after removing the second element we obtain the optimal answer — 4.

C. Wizards' Duel

2 seconds, 256 megabytes

Harry Potter and He-Who-Must-Not-Be-Named engaged in a fight to the death once again. This time they are located at opposite ends of the corridor of length l. Two opponents simultaneously charge a deadly spell in the enemy. We know that the impulse of Harry's magic spell flies at a speed of p meters per second, and the impulse of You-Know-Who's magic spell flies at a speed of q meters per second.

The impulses are moving through the corridor toward each other, and at the time of the collision they turn round and fly back to those who cast them without changing their original speeds. Then, as soon as the impulse gets back to it's caster, the wizard reflects it and sends again towards the enemy, without changing the original speed of the impulse.

Since Harry has perfectly mastered the basics of magic, he knows that after the second collision both impulses will disappear, and a powerful explosion will occur exactly in the place of their collision. However, the young wizard isn't good at math, so he asks you to calculate the distance from his position to the place of the second meeting of the spell impulses, provided that the opponents do not change positions during the whole fight.

Input

The first line of the input contains a single integer l $(1 \le l \le 1\,000)$ — the length of the corridor where the fight takes place.

The second line contains integer p, the third line contains integer q $(1 \le p, q \le 500)$ — the speeds of magical impulses for Harry Potter and He-Who-Must-Not-Be-Named, respectively.

Output

Print a single real number — the distance from the end of the corridor, where Harry is located, to the place of the second meeting of the spell impulses. Your answer will be considered correct if its absolute or relative error will not exceed 10^{-4}.

Namely: let's assume that your answer equals a, and the answer of the jury is b. The checker program will consider your answer correct if $\frac{|a-b|}{\max(1,b)} \le 10^{-4}$.

input
```
100
50
50
```
output
```
50
```

input
```
199
60
40
```
output
```
119.4
```

In the first sample the speeds of the impulses are equal, so both of their meetings occur exactly in the middle of the corridor.

D. Erase First or Second Letter

1 second, 256 megabytes

You are given a string s of length n. Let's define two operations you can apply on the string:

- remove the first character of the string;
- remove the second character of the string.

Your task is to find the number of distinct **non-empty** strings that can be generated by applying the given operations on the initial string any number of times (possibly zero), in any order.

Input

Each test consists of multiple test cases. The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains n $(1 \le n \le 10^5)$ — the length of the string.

The second line of each test case contains the string s. It is guaranteed that the string only contains lowercase letters of the English alphabet.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the number of distinct non-empty strings you can get.

input
```
5
5
aaaaa
1
z
5
ababa
14
bcdaaaabcdaaaa
20
abcdefghijklmnopqrst
```
output
```
5
1
9
50
210
```

In the first test case, we can get the following strings: a, aa, aaa, $aaaa$, $aaaaa$.

In the third test case, for example, the word ba can be reached in the following way:

- remove the first character of the current string $ababa$, getting $baba$;
- remove the second character of the current string $baba$, getting bba;
- remove the second character of the current string bba, getting ba.

E. Weight of the System of Nested Segments

2 seconds, 256 megabytes

On the number line there are m points, i-th of which has integer coordinate x_i and integer weight w_i. The coordinates of all points are different, and the points are numbered from 1 to m.

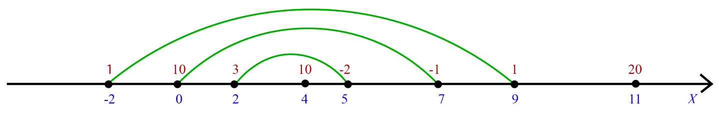
A sequence of n segments $[l_1, r_1], [l_2, r_2], \dots, [l_n, r_n]$ is called *system of nested segments* if for each pair i, j $(1 \le i < j \le n)$ the condition $l_i < l_j < r_j < r_i$ is satisfied. In other words, the second segment is strictly inside the first one, the third segment is strictly inside the second one, and so on.

For a given number n, find a system of nested segments such that:

- both ends of each segment are one of m given points;
- the sum of the weights $2 \cdot n$ of the points used as ends of the segments is **minimal**.

For example, let $m = 8$. The given points are marked in the picture, their weights are marked in red, their coordinates are marked in blue. Make a system of three nested segments:

- weight of the first segment: $1 + 1 = 2$
- weight of the second segment: $10 + (-1) = 9$
- weight of the third segment: $3 + (-2) = 1$
- sum of the weights of all the segments in the system: $2 + 9 + 1 = 12$



System of three nested segments

Input

The first line of input data contains an integer t $(1 \le t \le 10^4)$ —the number of input test cases.

An empty line is written before each test case.

The first line of each test case contains two positive integers n ($1 \le n \le 10^5$) and m ($2 \cdot n \le m \le 2 \cdot 10^5$).

The next m lines contain pairs of integers x_i ($-10^9 \le x_i \le 10^9$) and w_i ($-10^4 \le w_i \le 10^4$) — coordinate and weight of point number i ($1 \le i \le m$) respectively. All x_i are different.

It is guaranteed that the sum of m values over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output $n + 1$ lines: in the first of them, output the weight of the composed system, and in the next n lines output exactly two numbers — the indices of the points which are the endpoints of the i-th segment ($1 \le i \le n$). The order in which you output the endpoints of a segment is not important — you can output the index of the left endpoint first and then the number of the right endpoint, or the other way around.

If there are several ways to make a system of nested segments with minimal weight, output any of them.

input

```
3

3 8
0 10
-2 1
4 10
11 20
7 -1
9 1
2 3
5 -2

3 6
-1 2
1 3
3 -1
2 4
4 0
8 2

2 5
5 -1
3 -2
1 0
-2 0
-5 -3
```

output

```
12
2 6
5 1
7 8

10
1 6
5 2
3 4

-6
5 1
4 2
```

The first test case coincides with the example from the condition. It can be shown that the weight of the composed system is minimal.

The second test case has only 6 points, so you need to use each of them to compose 3 segments.

F. Card Game

2 seconds, 256 megabytes

Two players are playing an online card game. The game is played using a 32-card deck. Each card has a suit and a rank. There are four suits: clubs, diamonds, hearts, and spades. We will encode them with characters '`C`', '`D`', '`H`', and '`S`', respectively. And there are 8 ranks, in increasing order: '`2`', '`3`', '`4`', '`5`', '`6`', '`7`', '`8`', '`9`'.

Each card is denoted by two letters: its rank and its suit. For example, the 8 of Hearts is denoted as `8H`.

At the beginning of the game, one suit is chosen as the **trump suit**.

In each round, players make moves like this: the first player places one of his cards on the table, and the second player must beat this card with one of their cards. After that, both cards are moved to the discard pile.

A card can beat another card if both cards have the same suit and the first card has a higher rank than the second. For example, `8S` can beat `4S`. Additionally, a trump card can beat any non-trump card, regardless of the rank of the cards, for example, if the trump suit is clubs ('`C`'), then `3C` can beat `9D`. Note that trump cards can be beaten only by the trump cards of higher rank.

There were n rounds played in the game, so the discard pile now contains $2n$ cards. You want to reconstruct the rounds played in the game, but the cards in the discard pile are shuffled. Find any possible sequence of n rounds that might have been played in the game.

Input

The first line contains integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The first line of a test case contains the integer number n ($1 \le n \le 16$).

The second line of a test case contains one character, the trump suit. It is one of "`CDHS`".

The third line of a test case contains the description of $2n$ cards. Each card is described by a two-character string, the first character is the rank of the card, which is one of "`23456789`", and the second one is the suit of the card, which is one of "`CDHS`". All cards are different.

Output

For each test case print the answer to it:

- Print n lines. In each line, print the description of two cards, in the same format as in the input: the first card that was played by the first player, and then the card that was used by the second player to beat it.
- If there is no solution, print a single line "`IMPOSSIBLE`".

If there are multiple solutions, print any of them.

input

```
8
3
S
3C 9S 4C 6D 3S 7S
2
C
3S 5D 9S 6H
1
H
6C 5D
1
S
7S 3S
1
H
9S 9H
1
S
9S 9H
1
C
9D 8H
2
C
9C 9S 6H 8C
```

output

```
3C 4C
6D 9S
3S 7S
IMPOSSIBLE
IMPOSSIBLE
3S 7S
9S 9H
9H 9S
IMPOSSIBLE
6H 9C
9S 8C
```