A. City Day

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

For years, the Day of city N was held in the most rainy day of summer. New mayor decided to break this tradition and select a *not-so-rainy* day for the celebration. The mayor knows the weather forecast for the n days of summer. On the i-th day, a_i millimeters of rain will fall. All values a_i are distinct.

The mayor knows that citizens will watch the weather x days before the celebration and y days after. Because of that, he says that a day d is *not-so-rainy* if a_d is smaller than rain amounts at each of x days before day d and and each of y days after day d. In other words, $a_d < a_j$ should hold for all $d - x \le j < d$ and $d < j \le d + y$. Citizens only watch the weather during summer, so we only consider such j that $1 \le j \le n$.

Help mayor find the **earliest** *not-so-rainy* day of summer.

Input

The first line contains three integers n, x and y ($1 \le n \le 100\,000$, $0 \le x, y \le 7$) — the number of days in summer, the number of days citizens watch the weather before the celebration and the number of days they do that after.

The second line contains n distinct integers a_1, a_2, ..., a_n ($1 \le a_i \le 10^9$), where a_i denotes the rain amount on the i-th day.

Output

Print a single integer — the index of the earliest *not-so-rainy* day of summer. We can show that the answer always exists.

Examples

input

```
10 2 2
10 9 6 7 8 3 2 1 4 5
```

output

```
3
```

input

```
10 2 3
10 9 6 7 8 3 2 1 4 5
```

output

```
8
```

input

```
5 5 5
100000 10000 1000 100 10
```

output

```
5
```

Note

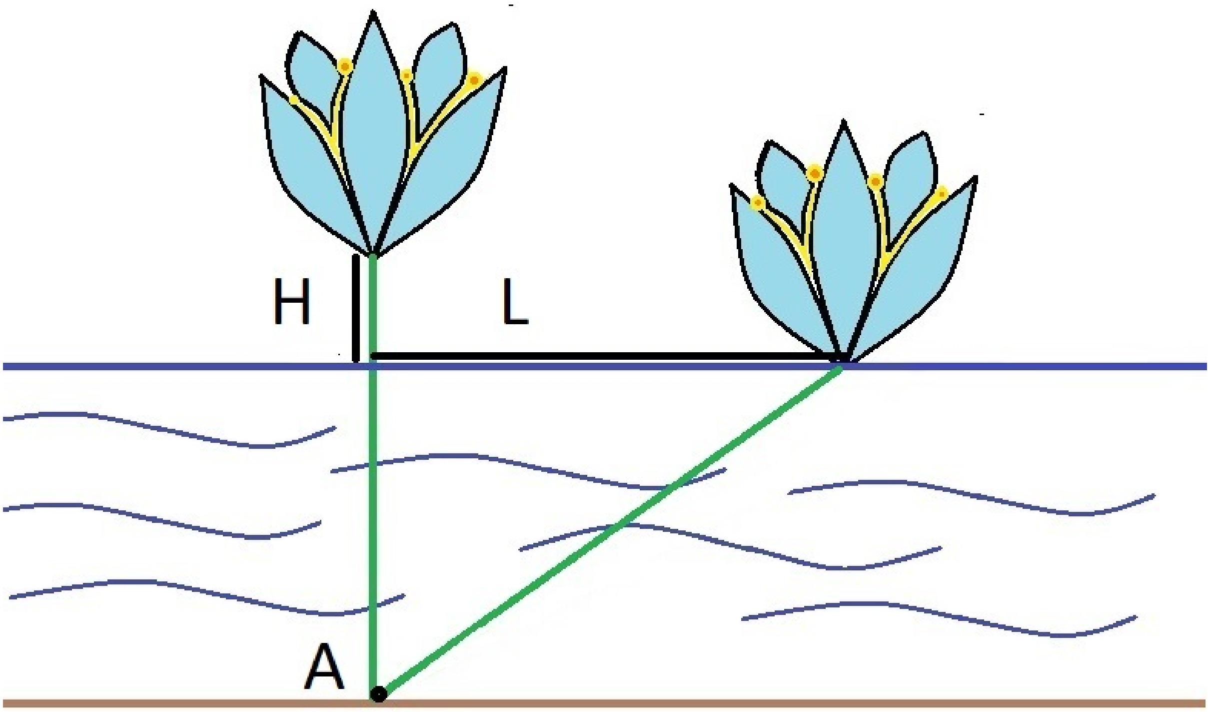
In the first example days 3 and 8 are *not-so-rainy*. The 3-rd day is earlier.

In the second example day 3 is not *not-so-rainy*, because $3 + y = 6$ and $a_3 > a_6$. Thus, day 8 is the answer. Note that $8 + y = 11$, but we don't consider day 11, because it is not summer.

B. Water Lily

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

While sailing on a boat, Inessa noticed a beautiful water lily flower above the lake's surface. She came closer and it turned out that the lily was exactly H centimeters above the water surface. Inessa grabbed the flower and sailed the distance of L centimeters. Exactly at this point the flower touched the water surface.



Suppose that the lily grows at some point A on the lake bottom, and its stem is always a straight segment with one endpoint at point A. Also suppose that initially the flower was exactly above the point A, i.e. its stem was vertical. Can you determine the depth of the lake at point A?

Input

The only line contains two integers H and L ($1 \le H < L \le 10^6$).

Output

Print a single number — the depth of the lake at point A. The absolute or relative error should not exceed 10^{-6}.

Formally, let your answer be A, and the jury's answer be B. Your answer is accepted if and only if $\frac{|A-B|}{\max(1,|B|)} \le 10^{-6}$.

Examples

input
```
1 2
```
output
```
1.5000000000000
```

input
```
3 5
```
output
```
2.6666666666667
```

C. MP3

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

One common way of digitalizing sound is to record sound intensity at particular time moments. For each time moment intensity is recorded as a non-negative integer. Thus we can represent a sound file as an array of n non-negative integers.

If there are exactly K distinct values in the array, then we need $k = \lceil \log_2 K \rceil$ bits to store each value. It then takes nk bits to store the whole file.

To reduce the memory consumption we need to apply some compression. One common way is to reduce the number of possible intensity values. We choose two integers $l \le r$, and after that all intensity values are changed in the following way: if the intensity value is within the range $[l;r]$, we don't change it. If it is less than l, we change it to l; if it is greater than r, we change it to r. You can see that we lose some low and some high intensities.

Your task is to apply this compression in such a way that the file fits onto a disk of size I bytes, and the number of changed elements in the array is minimal possible.

We remind you that 1 byte contains 8 bits.

$k = \lceil log_2 K \rceil$ is the smallest integer such that $K \le 2^k$. In particular, if $K = 1$, then $k = 0$.

Input

The first line contains two integers n and I ($1 \le n \le 4 \cdot 10^5$, $1 \le I \le 10^8$) — the length of the array and the size of the disk in bytes, respectively.

The next line contains n integers a_i ($0 \le a_i \le 10^9$) — the array denoting the sound file.

Output

Print a single integer — the minimal possible number of changed elements.

Examples

input

```
6 1
2 1 2 3 4 3
```

output

```
2
```

input

```
6 2
2 1 2 3 4 3
```

output

```
0
```

input

```
6 1
1 1 2 2 3 3
```

output

```
2
```

Note

In the first example we can choose $l = 2, r = 3$. The array becomes 2 2 2 3 3 3, the number of distinct elements is $K = 2$, and the sound file fits onto the disk. Only two values are changed.

In the second example the disk is larger, so the initial file fits it and no changes are required.

In the third example we have to change both 1s or both 3s.

D. Welfare State

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a country with n citizens. The i-th of them initially has a_i money. The government strictly controls the wealth of its citizens. Whenever a citizen makes a purchase or earns some money, they must send a receipt to the social services mentioning the amount of money they currently have.

Sometimes the government makes payouts to the poor: all citizens who have strictly less money than x are paid accordingly so that after the payout they have exactly x money. In this case the citizens don't send a receipt.

You know the initial wealth of every citizen and the log of all events: receipts and payouts. Restore the amount of money each citizen has after all events.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the numer of citizens.

The next line contains n integers a_1, a_2, ..., a_n ($0 \le a_i \le 10^9$) — the initial balances of citizens.

The next line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of events.

Each of the next q lines contains a single event. The events are given in chronological order.

Each event is described as either 1 p x ($1 \le p \le n$, $0 \le x \le 10^9$), or 2 x ($0 \le x \le 10^9$). In the first case we have a receipt that the balance of the p-th person becomes equal to x. In the second case we have a payoff with parameter x.

Output

Print n integers — the balances of all citizens after all events.

Examples

input

```
4
1 2 3 4
3
2 3
1 2 2
2 1
```

output

```
3 2 3 4
```

input

```
5
3 50 2 1 10
3
1 2 0
2 8
1 3 20
```

output

```
8 8 20 8 10
```

Note

In the first example the balances change as follows: `1 2 3 4` $\rightarrow$ `3 3 3 4` $\rightarrow$ `3 2 3 4` $\rightarrow$ `3 2 3 4`

In the second example the balances change as follows: `3 50 2 1 10` $\rightarrow$ `3 0 2 1 10` $\rightarrow$ `8 8 8 8 10` $\rightarrow$ `8 8 20 8 10`

E. Matching vs Independent Set

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a graph with $3 \cdot n$ vertices and m edges. You are to find a matching of n edges, **or** an independent set of n vertices.

A set of edges is called a matching if no two edges share an endpoint.

A set of vertices is called an independent set if no two vertices are connected with an edge.

Input

The first line contains a single integer $T \ge 1$ — the number of graphs you need to process. The description of T graphs follows.

The first line of description of a single graph contains two integers n and m, where $3 \cdot n$ is the number of vertices, and m is the number of edges in the graph ($1 \le n \le 10^5$, $0 \le m \le 5 \cdot 10^5$).

Each of the next m lines contains two integers v_i and u_i ($1 \le v_i, u_i \le 3 \cdot n$), meaning that there is an edge between vertices v_i and u_i.

It is guaranteed that there are no self-loops and no multiple edges in the graph.

It is guaranteed that the sum of all n over all graphs in a single test does not exceed 10^5, and the sum of all m over all graphs in a single test does not exceed $5 \cdot 10^5$.

Output

Print your answer for each of the T graphs. Output your answer for a single graph in the following format.

If you found a matching of size n, on the first line print "`Matching`" (without quotes), and on the second line print n integers — the indices of the edges in the matching. The edges are numbered from 1 to m in the input order.

If you found an independent set of size n, on the first line print "`IndSet`" (without quotes), and on the second line print n integers — the indices of the vertices in the independent set.

If there is no matching and no independent set of the specified size, print "`Impossible`" (without quotes).

You can print edges and vertices in any order.

If there are several solutions, print any. In particular, if there are both a matching of size n, and an independent set of size n, then you should print exactly one of such matchings **or** exactly one of such independent sets.

Example

input

```
4
1 2
1 3
1 2
1 2
1 3
1 2
2 5
1 2
3 1
```

```
1 4
5 1
1 6
2 15
1 2
1 3
1 4
1 5
1 6
2 3
2 4
2 5
2 6
3 4
3 5
3 6
4 5
4 6
5 6
```

output

```
Matching
2
IndSet
1
IndSet
2 4
Matching
1 15
```

Note

The first two graphs are same, and there are both a matching of size 1 and an independent set of size 1. Any of these matchings and independent sets is a correct answer.

The third graph does not have a matching of size 2, however, there is an independent set of size 2. Moreover, there is an independent set of size 5: 2 3 4 5 6. However such answer is not correct, because you are asked to find an independent set (or matching) of size **exactly** n.

The fourth graph does not have an independent set of size 2, but there is a matching of size 2.

F. Rectangle Painting 1

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a square grid of size $n \times n$. Some cells are colored in black, all others are colored in white. In one operation you can select some rectangle and color all its cells in white. It costs $\max(h, w)$ to color a rectangle of size $h \times w$. You are to make all cells white for minimum total cost.

Input

The first line contains a single integer n ($1 \leq n \leq 50$) — the size of the square grid.

Each of the next n lines contains a string of length n, consisting of characters '.' and '#'. The j-th character of the i-th line is '#' if the cell with coordinates (i, j) is black, otherwise it is white.

Output

Print a single integer — the minimum total cost to paint all cells in white.

Examples

input

```
3
###
#.#
###
```

output

```
3
```

input

```
3
...
...
...
```

output

```
0
```

input

```
4
```

```
#...
....
....
#...
```

output

```
2
```

input

```
5
#...#
.#.#.
.....
.#...
#....
```

output

```
5
```

Note

The examples and some of optimal solutions are shown on the pictures below.