A. Playing with Dice

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

Two players are playing a game. First each of them writes an integer from 1 to 6, and then a dice is thrown. The player whose written number got closer to the number on the dice wins. If both payers have the same difference, it's a draw.

The first player wrote number a, the second player wrote number b. How many ways to throw a dice are there, at which the first player wins, or there is a draw, or the second player wins?

Input

The single line contains two integers a and b ($1 \le a, b \le 6$) — the numbers written on the paper by the first and second player, correspondingly.

Output

Print three integers: the number of ways to throw the dice at which the first player wins, the game ends with a draw or the second player wins, correspondingly.

Examples

input
2 5
output
3 0 3

input
2 4
output
2 1 3

Note

The dice is a standard cube-shaped six-sided object with each side containing a number from 1 to 6, and where all numbers on all sides are distinct.

You can assume that number a is closer to number x than number b, if $|a - x| < |b - x|$.

B. Semifinals

Two semifinals have just been in the running tournament. Each semifinal had n participants. There are n participants advancing to the finals, they are chosen as follows: from each semifinal, we choose k people ($0 \le 2k \le n$) who showed the best result in their semifinals and all other places in the finals go to the people who haven't ranked in the top k in their semifinal but got to the $n - 2k$ of the best among the others.

The tournament organizers hasn't yet determined the k value, so the participants want to know who else has any chance to get to the finals and who can go home.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of participants in each semifinal.

Each of the next n lines contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$) — the results of the i-th participant (the number of milliseconds he needs to cover the semifinals distance) of the first and second semifinals, correspondingly. All results are distinct. Sequences $a_1, a_2, ..., a_n$ and $b_1, b_2, ..., b_n$ are sorted in ascending order, i.e. in the order the participants finished in the corresponding semifinal.

Output

Print two strings consisting of n characters, each equals either "0" or "1". The first line should correspond to the participants of the first semifinal, the second line should correspond to the participants of the second semifinal. The i-th character in the j-th line should equal "1" if the i-th participant of the j-th semifinal has any chances to advance to the finals, otherwise it should equal a "0".

Examples

input
```
4
9840 9920
9860 9980
9930 10020
10040 10090
```
output
```
1110
1100
```

input
```
4
9900 9850
9940 9930
10000 10020
10060 10110
```
output
```
1100
1100
```

Note

Consider the first sample. Each semifinal has 4 participants. The results of the first semifinal are 9840, 9860, 9930, 10040. The results of the second semifinal are 9920, 9980, 10020, 10090.

- If $k = 0$, the finalists are determined by the time only, so players 9840, 9860, 9920 and 9930 advance to the finals.
- If $k = 1$, the winners from both semifinals move to the finals (with results 9840 and 9920), and the other places are determined by the time (these places go to the sportsmen who run the distance in 9860 and 9930 milliseconds).
- If $k = 2$, then first and second places advance from each seminfial, these are participants with results 9840, 9860, 9920 and 9980 milliseconds.

C. Maze

Pavel loves grid mazes. A grid maze is an $n \times m$ rectangle maze where each cell is either empty, or is a wall. You can go from one cell to another only if both cells are empty and have a common side.

Pavel drew a grid maze with all empty cells forming a connected area. That is, you can go from any empty cell to any other one. Pavel doesn't like it when his maze has too little walls. He wants to turn exactly k empty cells into walls so that all the remaining cells still formed a connected area. Help him.

Input

The first line contains three integers n, m, k ($1 \le n, m \le 500$, $0 \le k < s$), where n and m are the maze's height and width, correspondingly, k is the number of walls Pavel wants to add and letter s represents the number of empty cells in the original maze.

Each of the next n lines contains m characters. They describe the original maze. If a character on a line equals ".", then the corresponding cell is empty and if the character equals "#", then the cell is a wall.

Output

Print n lines containing m characters each: the new maze that fits Pavel's requirements. Mark the empty cells that you transformed into walls as "X", the other cells must be left without changes (that is, "." and "#").

It is guaranteed that a solution exists. If there are multiple solutions you can output any of them.

Examples

input

```
3 4 2
#..#
..#.
#...
```

output

```
#.X#
X.#.
#...
```

input

```
5 4 5
#...
#.#.
.#..
...#
.#.#
```

output

```
#XXX
#X#.
X#..
...#
.#.#
```

D. Preparing for the Contest

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

Soon there will be held the world's largest programming contest, but the testing system still has m bugs. The contest organizer, a well-known university, has no choice but to attract university students to fix all the bugs. The university has n students able to perform such work. The students realize that they are the only hope of the organizers, so they don't want to work for free: the i-th student wants to get c_i 'passes' in his subjects (regardless of the volume of his work).

Bugs, like students, are not the same: every bug is characterized by complexity a_j, and every student has the level of his abilities b_i. Student i can fix a bug j only if the level of his abilities is not less than the complexity of the bug: $b_i \geq a_j$, and he does it in one day. Otherwise, the bug will have to be fixed by another student. Of course, no student can work on a few bugs in one day. All bugs are not dependent on each other, so they can be corrected in any order, and different students can work simultaneously.

The university wants to fix all the bugs as quickly as possible, but giving the students the total of not more than s passes. Determine which students to use for that and come up with the schedule of work saying which student should fix which bug.

Input

The first line contains three space-separated integers: n, m and s ($1 \leq n, m \leq 10^5, 0 \leq s \leq 10^9$) — the number of students, the number of bugs in the system and the maximum number of passes the university is ready to give the students.

The next line contains m space-separated integers $a_1, a_2, ..., a_m$ ($1 \leq a_i \leq 10^9$) — the bugs' complexities.

The next line contains n space-separated integers $b_1, b_2, ..., b_n$ ($1 \leq b_i \leq 10^9$) — the levels of the students' abilities.

The next line contains n space-separated integers $c_1, c_2, ..., c_n$ ($0 \leq c_i \leq 10^9$) — the numbers of the passes the students want to get for their help.

Output

If the university can't correct all bugs print "NO".

Otherwise, on the first line print "YES", and on the next line print m space-separated integers: the i-th of these numbers should equal the number of the student who corrects the i-th bug in the optimal answer. The bugs should be corrected as quickly as possible (you must spend the minimum number of days), and the total given passes mustn't exceed s. If there are multiple optimal answers, you can output any of them.

Examples

input
3 4 9 1 3 1 2 2 1 3 4 3 6
output
YES 2 3 2 3

input
3 4 10 2 3 1 2 2 1 3 4 3 6
output
YES 1 3 1 3

input
3 4 9 2 3 1 2 2 1 3 4 3 6
output
YES 3 3 2 3

input
3 4 5

```
1 3 1 2
2 1 3
5 3 6
```

output

NO

Note

Consider the first sample.

The third student (with level 3) must fix the 2nd and 4th bugs (complexities 3 and 2 correspondingly) and the second student (with level 1) must fix the 1st and 3rd bugs (their complexity also equals 1). Fixing each bug takes one day for each student, so it takes 2 days to fix all bugs (the students can work in parallel).

The second student wants 3 passes for his assistance, the third student wants 6 passes. It meets the university's capabilities as it is ready to give at most 9 passes.

E. Captains Mode

Kostya is a progamer specializing in the discipline of Dota 2. Valve Corporation, the developer of this game, has recently released a new patch which turned the balance of the game upside down. Kostya, as the captain of the team, realizes that the greatest responsibility lies on him, so he wants to resort to the analysis of innovations patch from the mathematical point of view to choose the best heroes for his team in every game.

A Dota 2 match involves two teams, each of them must choose some heroes that the players of the team are going to play for, and it is forbidden to choose the same hero several times, even in different teams. In large electronic sports competitions where Kostya's team is going to participate, the matches are held in the Captains Mode. In this mode the captains select the heroes by making one of two possible actions in a certain, predetermined order: pick or ban.

- To pick a hero for the team. After the captain picks, the picked hero goes to his team (later one of a team members will play it) and can no longer be selected by any of the teams.
- To ban a hero. After the ban the hero is not sent to any of the teams, but it still can no longer be selected by any of the teams.

The team captain may miss a pick or a ban. If he misses a pick, a random hero is added to his team from those that were available at that moment, and if he misses a ban, no hero is banned, as if there was no ban.

Kostya has already identified the strength of all the heroes based on the new patch fixes. Of course, Kostya knows the order of picks and bans. The strength of a team is the sum of the strengths of the team's heroes and both teams that participate in the match seek to maximize the difference in strengths in their favor. Help Kostya determine what team, the first one or the second one, has advantage in the match, and how large the advantage is.

Input

The first line contains a single integer n ($2 \le n \le 100$) — the number of heroes in Dota 2.

The second line contains n integers $s_1, s_2, ..., s_n$ ($1 \le s_i \le 10^6$) — the strengths of all the heroes.

The third line contains a single integer m ($2 \le m \le min(n, 20)$) — the number of actions the captains of the team must perform.

Next m lines look like "$action\ team$", where $action$ is the needed action: a pick (represented as a "p") or a ban (represented as a "b"), and $team$ is the number of the team that needs to perform the action (number 1 or 2).

It is guaranteed that each team makes at least one pick. Besides, each team has the same number of picks and the same number of bans.

Output

Print a single integer — the difference between the strength of the first team and the strength of the second team if the captains of both teams will act optimally well.

Examples

input
2 2 1 2 p 1 p 2
output
1

input
6 6 4 5 4 5 5 4 b 2 p 1 b 1 p 2
output
0

input
4 1 2 3 4 4 p 2

```
b 2
p 1
b 1
```

output

```
-2
```