Problem A. 810975

Time limit	1000 ms
Mem limit	262144 kB
OS	Windows

In the platypus kingdom, there is a popular game throughout the kingdom–Hearthstone. Hearthstone consists of eight players in each game, and only one of them wins. Due to the difficulty of winning, many platypuses share their best moments of playing the game via live streams. Recently, a Bible "810975" spread.

"810975, why can't newcomers understand?!"

"810975, why are you the only one who doesn't understand?"

...

Recently, Zayin is reciting the Bible 810975 every day. He also continues to explain to his friends the specific meaning of 810975: On August 10, the Platypus King played 9 games and won 7 times, including a 5-game winning streak. But today, there comes a new platypus, Ziyin, who doesn't know the meaning of 810975. So Zayin excitedly explains to him. However, Ziyin has a question, "How many situations are there if I play n games and win m games, and the longest winning streak is k?"

If we use 1 to represent victory and 0 to represent defeat, any string consisting of 0 and 1 of length n can represent a situation of an n-round game. Two situations are different if and only if the two 01 strings are different.

Input

The first line contains three integers n, m, k ($0 \leq n, m, k \leq 10^5$).

Output

The only line contains an integer: the number of situations meeting the constraints. You should output the answer modulo 998244353.

Examples

Input	Output
9 7 5	9

Problem B. Robot

Time limit	5000 ms
Mem limit	262144 kB
OS	Windows

Xiaochun, who has just started middle school, has a special fondness for robots. This weekend, she registered for the school's 'Little Robot Club'. After passing several rounds of exams, she finally managed to join the robot club.

Now, Xiaochun has placed her robot on an $n \times n$ grid, where both rows and columns are numbered from 1 to n. She has pre-programmed a sequence of instructions with a length of m in her remote control. These instructions include `U`,`D`,`L`,`R`:

- `U`: Move from (x, y) to $(x-1, y)$.
- `D`: Move from (x, y) to $(x+1, y)$.
- `L`: Move from (x, y) to $(x, y-1)$.
- `R`: Move from (x, y) to $(x, y+1)$.

Specifically, if the robot attempts to move outside the grid, that instruction is invalid.

Xiaochun has conducted a series of experiments, which fall into two categories:

1. Running experiments: She provides a coordinate (x, y) to place the robot and specifies a range $[l, r]$, indicating that the instructions numbered from l to r should be executed in sequence.
2. Modification experiments: She gives a position x and an instruction c, which means replacing the instruction at the x-th position in the sequence with c.

For each of these experimental runs, Xiaochun wants to know where her robot will ultimately stop, allowing her to verify if her robot's operation is normal.

Input

The first line contains a positive integer T $(1 \leq T \leq 10)$, representing the number of sets of data.

For each set of data:

The first line consists of three integers, n, m, and k, indicating the map's side length, the length of the instruction sequence, and the number of experiments, respectively. $(1 \leq$

$n, m, k \leq 5 \times 10^5$)

The second line contains a string of length m, representing the instruction sequence, consisting only of the characters `UDLR`.

Following are k lines, each beginning with the integer op, which represents the type of experiment:

- If $op = 1$, then the line is followed by four integers x, y, l, r ($1 \leq x, y \leq n, 1 \leq l \leq r \leq m$)
- If $op = 2$, then the line is followed by an integer x ($1 \leq x \leq m$) and a character c.

For datasets where $max(n, m, k) \geq 100$, there will be no more than two such datasets.

Output

For each instruction where $op = 1$, output two numbers x and y to represent the final position of the robot.

Examples

Input	Output
1 5 5 4 RDRRD 1 5 3 1 5 1 5 1 4 5 2 1 L 1 2 3 1 5	5 5 5 2 4 4

Problem C. Natatorium

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Tomorrow Programming School Department of Arithmetics is going to build a new pool in the basement of their building. The pool will serve mainly for testing of innovative floating point arithmetic algorithms, especially their floating properties in different kinds of liquids. Occasionally, the members of the staff will be allowed to the pool as well. To make the pool optimally fit for human needs, the department has to decide on the pool dimensions. The pool will be rectangular, it must not be a square. Its exact surface area C is predetermined by the choice of the relevant algorithms to be tested there. The company they hired to build the pool offers the list of possible pool side lengths. They have studied the demand in detail and they guarantee that a pool with the given area can be built using the lengths in the list. However, the particular choice is up to the department.

As the presented list is relatively long, the department also hired a junior programmer who is going to find out which lengths from the list can be chosen. Before he started his job, the programmer had made interesting observations. First, all available pool side lengths are primes. Second, the surface area C of the pool is a product of two distinct primes. He thinks this information may be helpful to find appropriate side lengths reasonably quickly.

Be faster than the hired programmer and solve the problem first.

Input

The first input line contains integer C ($1 \leq C \leq 10^{18}$), the area of the pool surface. The second line contains integer M ($1 \leq M \leq 2 \cdot 10^5$), the number of pool side lengths offered by the company. The third line contains a list of space-separated unique primes $P_1, \ldots, P_M$ ($1 < P_i < 10^9$), representing the offered pool side lengths.

All measurements are expressed in the same units.

Output

Output a single line with two space-separated pool side lengths from the offered list, which

can be used to build the pool. The first side length has to be smaller than the second one.

Examples

Input	Output
15 5 7 2 5 11 3	3 5

Problem D. Wall

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Tomorrow Programming School is going to decorate the front wall in the entrance hall with a distinctive pattern based on a visually appealing output of an elementary cellular automaton. The designers are going to study the outputs of several cellular automata and choose the one they like the most.

Your task is to write a program that replicates the output of an automaton.

Now, we describe elementary cellular automata and how they work. An elementary cellular automaton is a system consisting of a row of adjacent cells, and a rewriting rule. Each cell is always in one of two states: 0 or 1. The sequence of states of all cells, in the order of cells in the row, is called a generation.

The automaton operates in cycles. In one cycle, the current generation is taken as input and a new generation is calculated by applying the automaton rule to each cell in the current generation. At the end of a cycle, the current generation is replaced by the new generation. Then, another iteration of the cycle can start. The cycle can be repeated for any number of times.

It is assumed that the leftmost and the rightmost cells are also adjacent to other unused cells, which are always in state 0. This assumption ensures the automaton rule can be applied in the same way to all cells in the row. The unused cells do not appear in any input or output.

The state of a particular cell in the new generation is determined by its own state and the state of its two adjacent cells in the current generation, and by the automaton rule.

Let us consider a triple of cells consisting of a particular cell C, and its left and right adjacent cell. There are $2 \cdot 2 \cdot 2 = 8$ possible states, in which this triplet can be in the current generation: 0 or 1 in the left adjacent cell, 0 or 1 in the cell C itself, 0 or 1 in the right adjacent cell. Explicitly, all possible states of this triple of cells may be written as a sequence S of eight binary numbers: $S = (111, 110, 101, 100, 011, 010, 001, 000)$. The middle digit denotes the state of C, and the first and the third digits denote the states of the left and the right adjacent cell to C, respectively.

The automaton rule assigns one bit, 0 or 1, to each of the eight binary numbers in S. The rule is then coded as an 8-bit vector of the bits assigned to binary numbers in S. The

sequence S itself is not coded in the automaton rule.

Application of the rule on cell C consists of identifying the binary number in S which corresponds to the states of C and its two adjacent cells in the current generation. The state of C in the next generation is determined by the bit value assigned by the automaton rule to that particular binary number.

There are $2^8 = 256$ different automatons, distinguished by their rule codes. In the input of this problem, the 8-bit vectors representing automaton rules are coded in decimal.

Input

The first line of the input contains two integers, R and K ($0 \le R \le 255, 1 \le K \le 200$), the automaton rule coded in decimal and the number of generations, respectively. The second line of the input defines the first generation of the automaton. The cell states are coded by characters, with "`.`" coding 0 and "`X`" coding 1. The row of cells in the first generation is coded as a single string without spaces. The width of the row is at least 1 cell and it does not exceed 250 cells.

Output

The output contains the following K generations produced by the given automaton. The generations are coded and formatted in the same way as the first generation in the input. Each generation occupies one line, there are no empty lines in the output.

Examples

Input	Output
128 5 XXXXXXXXXXXXX	.XXXXXXXXXX. ..XXXXXXXX.. ...XXXXXX... ....XXXX.... .....XX.....

Input	Output
30 10 ....X..........	..........XXX.......... .........XX..X......... ........XX.XXXX........ .......XX..X...X....... ......XX.XXXX.XXX...... .....XX..X....X..X..... ....XX.XXXX..XXXXXX.... ...XX..X...XXX.....X... ..XX.XXXX.XX..X...XXX.. .XX..X....X.XXXX.XX..X.

Problem E. Movers

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Department of successful computing in Tomorrow Programming School is famous for stockpiling large quantities of desks and monitors in their labs, it serves well to the whole community.

Any time a meeting is being held in a lab, the number of desks and monitors in it should be sufficient to serve all participants. In case of necessity, additional desks and/or monitors may be borrowed from neighbouring labs. In extreme cases, all desks and all monitors from all neighbouring labs may be brought in. Thus, available desks and monitors in a lab are exactly those desks and monitors which are in the lab itself and in all its neighbour labs. Desks or monitors are never transported from more distant labs, it is deemed ineffective and accident prone. After a meeting, all borrowed desks and monitors are returned back to their original labs, before any other meeting starts.

The desired configuration for a meeting is when the number of available desks and monitors in the lab is equal. Often, the number of available desks in a lab is either smaller or bigger than the number of available monitors, and that creates specific problems for the maintenance staff each time.

The inflow of desks and monitors to the department is rapid. Frequently a shipment of desks and monitors arrives to the department and its contents is added to various labs. It is added immediately or immediately after the end of the current meeting.

To plan the meetings, and equipment maintenance as well, it is important to know how many desks and monitors are available in any lab at any moment. The department needs a program that can process two kinds of queries. The first kind of query specifies a number of desks or monitors that have just been added to a particular lab. The second kind of query asks for a relation between the number of available desks and monitors in a particular lab.

Input

The first input line contains three integers N, M, Q ($1 \leq N \leq 10^5, 0 \leq M \leq 10^5, 0 \leq Q \leq 10^5$), the number of labs, the number of pairs of labs which are neighbours to each other, and the number of queries. Labs are labeled by integers $1, 2, \ldots, N$. The second line contains N spaceseparated integers D_i ($0 \leq D_i \leq 100$), the number of desks in lab i. The

third line contains N space-separated integers E_i ($0 \leq E_i \leq 100$), the number of monitors in lab i. Next M lines contain space-separated unique pairs of distinct integers a_i, b_i, ($1 \leq a_i, b_i \leq N$), the labels of pairs of neighbour labs. Next Q lines contain the queries, one query per line. Each query is provided in one of the two formats.

1. `add <count>` desk/monitor `<label>` – query increases the number of desks or monitors by the given `<count>` in a lab with specified `<label>`.
2. `check <label>` – query asks for a relation between the number of available desks and available monitors in a lab with specified `<label>`.

One add query increases the number of desks or monitors in a lab by at most 100.

Output

Output Q lines, one for each query of type check, in the order they appear in the input. Each line contains one of the strings "`desks`", "`monitors`", or `same`", depending on whether there are more available desks or more available monitors in the lab specified by the query, or whether their numbers are equal.

Examples

Input	Output
4 5 8 1 1 1 0 2 0 2 0 1 2 2 3 3 4 4 1 1 3 check 2 add 2 desk 1 check 2 add 1 monitor 3 check 1 check 2 check 3 check 4	monitors desks same same same monitors

Problem F. Screamers in the Storm

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

Students visiting the Multidimensional Data Cabinet at Tomorrow Programming School may get access to multidimensional hyperspheres stored in air-conditioned helium cupboards in the cabinet. Before exploring a chosen sphere, a student has to prove he/she is knowledgeable enough to handle the sphere. There are spheres of various dimensions and various integer radiuses, each in a separate locked cupboard. The dimension and the radius of the sphere are written on the cupboard label. To unlock the cupboard, the student has to enter a specific value into the lock mechanism. The value depends on the dimension and the radius of the sphere. It is equal to the sum of absolute values of all coordinates of all integer points which lie inside or on the surface of the sphere, when the center of the sphere is exactly in the center of coordinates. The sum has to be entered modulo $10^9 + 7$.

An integer point in a d-dimensional Euclidean space is a point which all d coordinates are integers.

The Cabinet manager is planning to obtain more spheres in the future, and you know that calculations which help to unlock some high-dimensional spheres may be quite tedious to perform by hand.

Write a program that calculates the value necessary to unlock a cupboard with a sphere in the cabinet.

Input

The input contains two integers D, R ($1 \leq D \leq 50, 1 \leq R \leq 50$), the dimension and the radius of the sphere inside a cupboard.

Output

Print the value which unlocks the cupboard with the given sphere, modulo $10^9 + 7$.

Examples

Input	Output
1 6	42

Input	Output
3 5	2850

Problem G. Digitalisation

Time limit	5000 ms
Mem limit	1048576 kB
OS	Windows

The students of Tomorrow Programming School are admitted to the school in the nationwide School Admission Process (SAP), which involves most schools in the region. SAP consists of two rounds.

At the beginning of the first SAP round, each student applies for two schools, while indicating his priorities. For each student there is a school which is his first priority and a school which is his second priority. All students who apply are tested in a national exam that gives each student a unique score and creates a common list CL of students sorted according to their scores in the test in descending order (the highest score is the best). CL is available to each school. Each school prefers to admit students with better score. The capacity of each school is bounded by common capacity limit C. Each school keeps a list of candidates, the capacity of the list is also C. The lists are initially empty.

Then, the second round of SAP starts. It consists of one or more cycles of so-called update events. In a cycle, each school, one after another, performs one update event.

In an update event, the school with update list L searches CL from its end (starting with the worst score applicant) and they select the first applicant A who is not in L and who fulfills all following conditions:

- Either L is not full, or the score of A is better than the worst score of the candidates in L.
- The school with list L is either the first or the second priority of A.
- A is currently either not on a candidate list of any school or they are currently on candidate list of their second priority school.

If the school cannot select an applicant fulfilling the conditions, the update event is claimed to be empty and it is terminated. Otherwise, if L is full at the moment, the candidate with worst score in L is removed from L. Next, A is included in L. If A had been to this moment on a candidate list of another school, A is removed from that list. Finally, the update event is claimed to be valid.

If, during a cycle, at least one update event is valid, a new cycle is started all over again. The second round of SAP ends when all update events in one cycle are empty. Then, the schools admit the students which are on their respective candidate lists. The regional statisticians

want to know, how many students were admitted to their first or second priority schools. Write an appropriate program for them.

Input

The first input line contains three integers N, M, C ($2 \leq N, M \leq 10^5; 1 \leq C \leq 100$), the total number of student applicants, the number of schools, and the common admission capacity.

Schools are labeled by integers $1, 2, \ldots, M$. Each of the following N lines contain the first and the second priority of one student, represented by the labels of the schools. The lines are sorted in descending order of the scores of the corresponding students.

Output

Output two numbers on a single line. The number of students who got to their desired school with first priority and the number of students who got to their desired school with second priority.

Examples

Input	Output
9 3 4 1 2 2 3 1 3 3 2 1 2 3 2 2 3 2 3 2 1	9 0

Input	Output
4 2 1 1 2 1 2 1 2 1 2	1 1