Problem A
Boarding Queue
Time limit: 2 seconds

You are on your way to compete in ICPC Asia Pacific Championship. Unfortunately, you are in the process of the worst part of flying: waiting in the boarding queue.

You are in a queue with n travelers, numbered from 1 to n ordered from the front of the queue to the back of the queue.

The boarding area is represented by a grid of r rows and c columns, where the rows are numbered from 1 to r (top to bottom) and the columns are numbered from 1 to c (left to right). Each traveler occupies exactly one distinct cell in the grid. Two travelers are *adjacent* if the cells they are in share an edge. Traveler t and traveler $t-1$ are guaranteed to be adjacent, for any $2 \le t \le n$.

For example, Figure A.1 illustrates a possible location of the travelers. In this example, traveler 1 is adjacent to travelers 2 and 10 but not adjacent to traveler 11.

11				
10	1	2		
9		3	4	
8	7	6	5	

Figure A.1: Example location of the travelers.

At each boarding step, all of the following happen simultaneously:

- The frontmost traveler in the queue, say traveler t, boards the aircraft and leaves the boarding area.
- For each t' ($t+1 \le t' \le n$), traveler t' takes the cell that traveler $t'-1$ was occupying immediately before the step.

For example, Figure A.2 illustrates the locations of the travelers after the first three boarding steps from the initial location above.

You are traveler p (that is, there are $p-1$ travelers in front of you). You know that your team coach is somewhere in the queue, but you do not know where. Assuming your team coach is equally likely to be any of travelers 1 to n (except p), you want to calculate the probability that you will be adjacent to your team coach at some point before you board the aircraft. Formally, you will be adjacent to traveler q at some point before you board the aircraft if there exists an integer s ($0 \le s < p$) such that traveler p and traveler q are adjacent after s boarding steps.

Input

The first line of input contains four integers r, c, n, and p ($1 \le r, c \le 1000; 2 \le n \le r \times c; 1 \le p \le n$). Each of the next r lines contains c integers. The j-th integer in the i-th line denotes $G_{i,j}$ ($0 \le G_{i,j} \le n$), where a non-zero

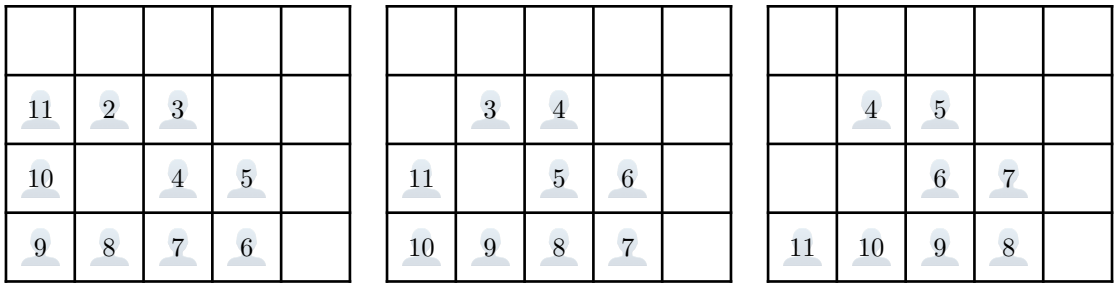


Figure A.2: Location of the travelers after 1, 2, and 3 boarding steps respectively.

value of $G_{i,j}$ means that traveler $G_{i,j}$ initially occupies the cell at row i and column j of the boarding area, while a zero value of $G_{i,j}$ means that no traveler occupies the cell. Across all pairs (i, j), each of the integers 1 to n appears exactly once in $G_{i,j}$. The input guarantees that traveler t and traveler $t-1$ are adjacent, for all $2 \leq t \leq n$.

Output

Output a fraction in the x/y format indicating the probability that you will be adjacent to your team coach at some point before you board the aircraft. The value of y must be equal to $n-1$. Note that there must not be spaces between the integers and the `/` delimiter.

Sample Input #1	Sample Output #1
4 5 11 2 11 0 0 0 0 10 1 2 0 0 9 0 3 4 0 8 7 6 5 0	3/10

Explanation for the sample input/output #1

This sample corresponds to the example given in the problem description above. You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 1, 3, or 11.

Sample Input #2	Sample Output #2
1 7 7 6 1 2 3 4 5 6 7	2/6

Explanation for the sample input/output #2

You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 5 or 7. Note that the output `1/3` is **not** accepted.

Dress Rehearsal Problem B
Combination Lock
Time limit: 2 seconds

Your house is protected by a combination lock containing n rotating discs, numbered from 1 to n. On a typical combination lock, each rotating disc has 10 symbols, represented by integers between 0 to 9, inclusive. Since you are a mathematician, your combination lock is not typical. Instead, each rotating disc on your combination lock may have a different number of symbols. In particular, rotating disc i has $b_i - a_i + 1$ symbols, represented by integers between a_i to b_i, inclusive.

The combination lock is unlocked when each rotating disc displays one integer, and any pair of two integers displayed by the rotating disc are coprime. Two integers are coprime if they do not have any common positive factors other than 1.

You want to unlock the combination lock, so you want to determine what integer to be displayed on each combination lock to satisfy the requirement above. It is possible that your combination lock was sabotaged when you were gone so it might be impossible to unlock your combination lock.

Input

The first line of input contains one integer n ($2 \leq n \leq 50$). Each of the next n lines contains two integers. The i-th line contains a_i and b_i ($1 \leq a_i \leq b_i \leq 50$).

Output

Output one line containing n integers, where the i-th integer represents the integer to be displayed by rotating disc i to unlock the combination lock. If there are multiple solutions, you can output any of them. If there is no solution, output just the integer `-1`.

Sample Input #1	Sample Output #1
4 3 9 2 8 1 4 2 10	3 7 1 10

Explanation for the sample input/output #1

Other solutions such as `9 5 2 7` are also accepted.

Sample Input #2	Sample Output #2
3 5 6 5 6 2 6	-1

This page is intentionally left blank.

Dress Rehearsal Problem C

Candy Compress

Time limit: 4 seconds

You are developing a mobile game called *Candy Compress*. In this game, there are several colored candies lined up from left to right. There are 26 possible colors. At any point in time, the player can choose to add a candy at any position or to remove a subset of neighboring candies to get points depending on the colors of the removed candies.

To develop the game, you need to implement the following data structure. Initially, the data structure loads a 1-indexed string of n characters, which represents the colors of the initial candies. The string consists of only uppercase Latin characters (`A`–`Z`). After loading the string, there are q operations that you need to support. Each operation is either one of the following:

Operation 1: Insert the uppercase Latin character c to the string so that the character is in the i-th position in the new string. In particular, $i = 1$ means inserting character c at the beginning of the string. It is guaranteed that $1 \leq i \leq m + 1$, where m is the length of the string just before this operation.

Operation 2: Remove the characters of the string from the l-th to the r-th position, inclusive. It is guaranteed that $1 \leq l \leq r \leq m$, where m is the length of the string just before this operation.

For each Operation 2, your data structure needs to determine the characters that are removed, so that the game can calculate the number of points to be given to the player. In other words, you need to determine the content of the string from the l-th position to the r-th position just before the operation.

Input

The first line of input contains two integers n and q ($1 \leq n \leq 500\,000$; $1 \leq q \leq 500\,000$). The second line contains a string consisting of n uppercase Latin characters representing the initial string to be loaded by the data structure. Each of the next q lines represents an operation with either one of the following formats:

1. `1` c i represents an Operation 1. It is guaranteed that c is an uppercase Latin character and $1 \leq i \leq m + 1$, where m is the length of the string just before this operation.
2. `2` l r represents an Operation 2. It is guaranteed that $1 \leq l \leq r \leq m$ where m is the length of the string just before this operation.

The operations are given in the order they are to be performed. It is guaranteed that there is at least one Operation 2.

Output

For each Operation 2 in order, output one line containing the characters that are removed by the operation.

Sample Input #1	Sample Output #1
3 5 XPA 1 P 3 1 P 5 2 2 5 1 Y 2 2 1 2	PPAP XY

Explanation for the sample input/output #1

The first and second operations modify the string in the data structure as follows: `XPA` $\rightarrow$ `XP`**`P`**`A` $\rightarrow$ `XPPA`**`P`**. The third operation removes `PPAP` from the string, leaving the string `X` in the data structure. The fourth operation inserts the character `Y` to the end of the string. The last operation removes `XY` from the string, leaving an empty string in the data structure.

Sample Input #2	Sample Output #2
27 7 ICPCASIAPACIFICCHAMPIONSHIP 2 5 8 2 5 11 1 A 5 1 P 6 1 A 7 1 C 8 2 1 8	ASIA PACIFIC ICPCAPAC

Problem D
Secret Lilies and Roses
Time limit: 2 seconds

There are n flowers arranged in a line from left to right, which are numbered from 1 to n in that order. Each flower is either a lily or a rose. For an integer j between 0 and n, inclusive, let l_j denote the number of lilies among the leftmost j flowers, and let r_j denote the number of roses among the rightmost $n - j$ flowers.

Initially, only the number of flowers n is provided to you. The types of the flowers are hidden. You can obtain information on the flowers by making *queries*. In one query, you can perform one of the following.

Type query: Specify an integer i between 1 and n, inclusive. You will then receive the type of flower i.

Multiply query: Specify an integer j between 0 and n, inclusive. You will then receive the value of $l_j \times r_j$.

Your task is to find an integer k between 0 and n, inclusive, for which $l_k = r_k$ by making a limited number of queries. You can assume that at least one such integer exists for the arrangement of the flower types. Note that you do not need to identify the type of each flower.

Interaction

The first line of input contains one integer t ($1 \leq t \leq 100$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line of each test case contains an integer n ($1 \leq n \leq 100$). After reading it, your program can start making queries. For each query, your program should write a line containing one of the following:

- "`type` i" to perform a Type query by specifying an integer i ($1 \leq i \leq n$). In response, an input line containing a string either `lily` or `rose` becomes available, indicating the type of flower i.
- "`multi` j" to perform a Multiply query by specifying an integer j ($0 \leq j \leq n$). In response, an input line containing the integer $l_j \times r_j$ becomes available.

Your program is allowed to make up to **10 queries** for each test case. This means the total number of Type queries and Multiply queries combined must not exceed 10. If your program makes more than 10 queries, it will be judged as "Wrong Answer".

When your program identifies an integer k such that $l_k = r_k$, it should write a line of the form "`answer` k" ($0 \leq k \leq n$). Note that providing the answer does not count as a query.

The input guarantees that at least one correct output exists. If there are multiple correct outputs, any one of them will be accepted.

After writing the answer line, your program should start processing the next test case. When all t test cases have been processed, the interaction stops and your program should terminate.

If your program is judged as "Wrong Answer", an input line containing `-1` becomes available. After that, the interaction stops without processing the remaining test cases, and your program should terminate.

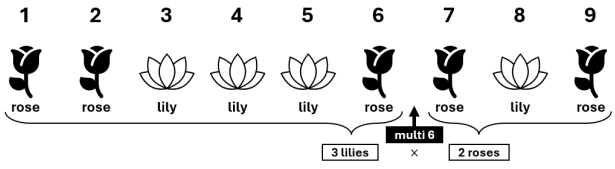
Notes on interactive judging:

- The evaluation is non-adversarial, meaning that the types of the flowers are chosen in advance rather than in response to your queries.
- Do not forget to flush output buffers after writing. See the "Judging Details" document for details.
- You are provided with a command-line tool for local testing, together with input files corresponding to the sample interactions. You can download these files from DOMjudge. The tool has comments at the top to explain its use.

Read	Sample Interaction #1	Write
2		
9		
		type 8
lily		
		type 1
rose		
		multi 6
6		
		multi 3
3		
		answer 5
3		
		multi 3
0		
		type 3
rose		
		answer 3

Explanation for the sample interaction #1

For the first test case, nine flowers are arranged as shown in the following figure. In response to the third query, $l_6 \times r_6 = 3 \times 2 = 6$ is returned, and in response to the fourth query, $l_3 \times r_3 = 1 \times 3 = 3$ is returned. Since $l_5 = r_5 = 3$, $k = 5$ is a correct output.



Problem E
Boarding Queue
Time limit: 2 seconds

You are on your way to compete in ICPC Asia Pacific Championship. Unfortunately, you are in the process of the worst part of flying: waiting in the boarding queue.

You are in a queue with n travelers, numbered from 1 to n ordered from the front of the queue to the back of the queue.

The boarding area is represented by a grid of r rows and c columns, where the rows are numbered from 1 to r (top to bottom) and the columns are numbered from 1 to c (left to right). Each traveler occupies exactly one distinct cell in the grid. Two travelers are *adjacent* if the cells they are in share an edge. Traveler t and traveler $t-1$ are guaranteed to be adjacent, for any $2 \le t \le n$.

For example, Figure E.1 illustrates a possible location of the travelers. In this example, traveler 1 is adjacent to travelers 2 and 10 but not adjacent to traveler 11.

11				
10	1	2		
9		3	4	
8	7	6	5	

Figure E.1: Example location of the travelers.

At each boarding step, all of the following happen simultaneously:

- The frontmost traveler in the queue, say traveler t, boards the aircraft and leaves the boarding area.
- For each t' ($t+1 \le t' \le n$), traveler t' takes the cell that traveler $t'-1$ was occupying immediately before the step.

For example, Figure E.2 illustrates the locations of the travelers after the first three boarding steps from the initial location above.

You are traveler p (that is, there are $p-1$ travelers in front of you). You know that your team coach is somewhere in the queue, but you do not know where. Assuming your team coach is equally likely to be any of travelers 1 to n (except p), you want to calculate the probability that you will be adjacent to your team coach at some point before you board the aircraft. Formally, you will be adjacent to traveler q at some point before you board the aircraft if there exists an integer s ($0 \le s < p$) such that traveler p and traveler q are adjacent after s boarding steps.

Input

The first line of input contains four integers r, c, n, and p ($1 \le r, c \le 1000; 2 \le n \le r \times c; 1 \le p \le n$). Each of the next r lines contains c integers. The j-th integer in the i-th line denotes $G_{i,j}$ ($0 \le G_{i,j} \le n$), where a non-zero

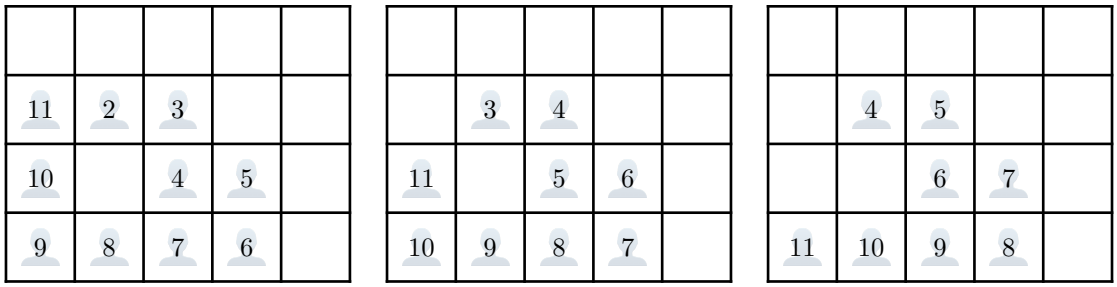


Figure E.2: Location of the travelers after 1, 2, and 3 boarding steps respectively.

value of $G_{i,j}$ means that traveler $G_{i,j}$ initially occupies the cell at row i and column j of the boarding area, while a zero value of $G_{i,j}$ means that no traveler occupies the cell. Across all pairs (i, j), each of the integers 1 to n appears exactly once in $G_{i,j}$. The input guarantees that traveler t and traveler $t-1$ are adjacent, for all $2 \leq t \leq n$.

Output

Output a fraction in the x/y format indicating the probability that you will be adjacent to your team coach at some point before you board the aircraft. The value of y must be equal to $n-1$. Note that there must not be spaces between the integers and the `/` delimiter.

Sample Input #1	Sample Output #1
4 5 11 2 11 0 0 0 0 10 1 2 0 0 9 0 3 4 0 8 7 6 5 0	3/10

Explanation for the sample input/output #1

This sample corresponds to the example given in the problem description above. You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 1, 3, or 11.

Sample Input #2	Sample Output #2
1 7 7 6 1 2 3 4 5 6 7	2/6

Explanation for the sample input/output #2

You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 5 or 7. Note that the output `1/3` is **not** accepted.

Dress Rehearsal Problem F
Combination Lock
Time limit: 2 seconds

Your house is protected by a combination lock containing n rotating discs, numbered from 1 to n. On a typical combination lock, each rotating disc has 10 symbols, represented by integers between 0 to 9, inclusive. Since you are a mathematician, your combination lock is not typical. Instead, each rotating disc on your combination lock may have a different number of symbols. In particular, rotating disc i has $b_i - a_i + 1$ symbols, represented by integers between a_i to b_i, inclusive.

The combination lock is unlocked when each rotating disc displays one integer, and any pair of two integers displayed by the rotating disc are coprime. Two integers are coprime if they do not have any common positive factors other than 1.

You want to unlock the combination lock, so you want to determine what integer to be displayed on each combination lock to satisfy the requirement above. It is possible that your combination lock was sabotaged when you were gone so it might be impossible to unlock your combination lock.

Input

The first line of input contains one integer n ($2 \leq n \leq 50$). Each of the next n lines contains two integers. The i-th line contains a_i and b_i ($1 \leq a_i \leq b_i \leq 50$).

Output

Output one line containing n integers, where the i-th integer represents the integer to be displayed by rotating disc i to unlock the combination lock. If there are multiple solutions, you can output any of them. If there is no solution, output just the integer `-1`.

Sample Input #1	Sample Output #1
4 3 9 2 8 1 4 2 10	3 7 1 10

Explanation for the sample input/output #1

Other solutions such as `9 5 2 7` are also accepted.

Sample Input #2	Sample Output #2
3 5 6 5 6 2 6	-1

This page is intentionally left blank.

Dress Rehearsal Problem G

Candy Compress

Time limit: 4 seconds

You are developing a mobile game called *Candy Compress*. In this game, there are several colored candies lined up from left to right. There are 26 possible colors. At any point in time, the player can choose to add a candy at any position or to remove a subset of neighboring candies to get points depending on the colors of the removed candies.

To develop the game, you need to implement the following data structure. Initially, the data structure loads a 1-indexed string of n characters, which represents the colors of the initial candies. The string consists of only uppercase Latin characters (`A`–`Z`). After loading the string, there are q operations that you need to support. Each operation is either one of the following:

Operation 1: Insert the uppercase Latin character c to the string so that the character is in the i-th position in the new string. In particular, $i = 1$ means inserting character c at the beginning of the string. It is guaranteed that $1 \leq i \leq m + 1$, where m is the length of the string just before this operation.

Operation 2: Remove the characters of the string from the l-th to the r-th position, inclusive. It is guaranteed that $1 \leq l \leq r \leq m$, where m is the length of the string just before this operation.

For each Operation 2, your data structure needs to determine the characters that are removed, so that the game can calculate the number of points to be given to the player. In other words, you need to determine the content of the string from the l-th position to the r-th position just before the operation.

Input

The first line of input contains two integers n and q ($1 \leq n \leq 500\,000$; $1 \leq q \leq 500\,000$). The second line contains a string consisting of n uppercase Latin characters representing the initial string to be loaded by the data structure. Each of the next q lines represents an operation with either one of the following formats:

1. `1` c i represents an Operation 1. It is guaranteed that c is an uppercase Latin character and $1 \leq i \leq m + 1$, where m is the length of the string just before this operation.
2. `2` l r represents an Operation 2. It is guaranteed that $1 \leq l \leq r \leq m$ where m is the length of the string just before this operation.

The operations are given in the order they are to be performed. It is guaranteed that there is at least one Operation 2.

Output

For each Operation 2 in order, output one line containing the characters that are removed by the operation.

Sample Input #1	Sample Output #1
3 5 XPA 1 P 3 1 P 5 2 2 5 1 Y 2 2 1 2	PPAP XY

Explanation for the sample input/output #1

The first and second operations modify the string in the data structure as follows: `XPA` $\rightarrow$ `XP`**`P`**`A` $\rightarrow$ `XPPA`**`P`**. The third operation removes `PPAP` from the string, leaving the string `X` in the data structure. The fourth operation inserts the character `Y` to the end of the string. The last operation removes `XY` from the string, leaving an empty string in the data structure.

Sample Input #2	Sample Output #2
27 7 ICPCASIAPACIFICCHAMPIONSHIP 2 5 8 2 5 11 1 A 5 1 P 6 1 A 7 1 C 8 2 1 8	ASIA PACIFIC ICPCAPAC

Problem H
Secret Lilies and Roses
Time limit: 2 seconds

There are n flowers arranged in a line from left to right, which are numbered from 1 to n in that order. Each flower is either a lily or a rose. For an integer j between 0 and n, inclusive, let l_j denote the number of lilies among the leftmost j flowers, and let r_j denote the number of roses among the rightmost $n - j$ flowers.

Initially, only the number of flowers n is provided to you. The types of the flowers are hidden. You can obtain information on the flowers by making *queries*. In one query, you can perform one of the following.

Type query: Specify an integer i between 1 and n, inclusive. You will then receive the type of flower i.

Multiply query: Specify an integer j between 0 and n, inclusive. You will then receive the value of $l_j \times r_j$.

Your task is to find an integer k between 0 and n, inclusive, for which $l_k = r_k$ by making a limited number of queries. You can assume that at least one such integer exists for the arrangement of the flower types. Note that you do not need to identify the type of each flower.

Interaction

The first line of input contains one integer t ($1 \leq t \leq 100$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line of each test case contains an integer n ($1 \leq n \leq 100$). After reading it, your program can start making queries. For each query, your program should write a line containing one of the following:

- "`type` i" to perform a Type query by specifying an integer i ($1 \leq i \leq n$). In response, an input line containing a string either `lily` or `rose` becomes available, indicating the type of flower i.
- "`multi` j" to perform a Multiply query by specifying an integer j ($0 \leq j \leq n$). In response, an input line containing the integer $l_j \times r_j$ becomes available.

Your program is allowed to make up to **10 queries** for each test case. This means the total number of Type queries and Multiply queries combined must not exceed 10. If your program makes more than 10 queries, it will be judged as "Wrong Answer".

When your program identifies an integer k such that $l_k = r_k$, it should write a line of the form "`answer` k" ($0 \leq k \leq n$). Note that providing the answer does not count as a query.

The input guarantees that at least one correct output exists. If there are multiple correct outputs, any one of them will be accepted.

After writing the answer line, your program should start processing the next test case. When all t test cases have been processed, the interaction stops and your program should terminate.

If your program is judged as "Wrong Answer", an input line containing -1 becomes available. After that, the interaction stops without processing the remaining test cases, and your program should terminate.

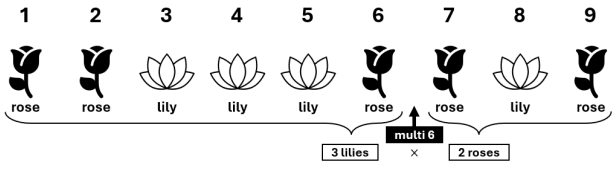
Notes on interactive judging:

- The evaluation is non-adversarial, meaning that the types of the flowers are chosen in advance rather than in response to your queries.
- Do not forget to flush output buffers after writing. See the "Judging Details" document for details.
- You are provided with a command-line tool for local testing, together with input files corresponding to the sample interactions. You can download these files from DOMjudge. The tool has comments at the top to explain its use.

Read	Sample Interaction #1	Write
2		
9		
		type 8
lily		
		type 1
rose		
		multi 6
6		
		multi 3
3		
		answer 5
3		
		multi 3
0		
		type 3
rose		
		answer 3

Explanation for the sample interaction #1

For the first test case, nine flowers are arranged as shown in the following figure. In response to the third query, $l_6 \times r_6 = 3 \times 2 = 6$ is returned, and in response to the fourth query, $l_3 \times r_3 = 1 \times 3 = 3$ is returned. Since $l_5 = r_5 = 3$, $k = 5$ is a correct output.



Problem I
Boarding Queue
Time limit: 2 seconds

You are on your way to compete in ICPC Asia Pacific Championship. Unfortunately, you are in the process of the worst part of flying: waiting in the boarding queue.

You are in a queue with n travelers, numbered from 1 to n ordered from the front of the queue to the back of the queue.

The boarding area is represented by a grid of r rows and c columns, where the rows are numbered from 1 to r (top to bottom) and the columns are numbered from 1 to c (left to right). Each traveler occupies exactly one distinct cell in the grid. Two travelers are *adjacent* if the cells they are in share an edge. Traveler t and traveler $t-1$ are guaranteed to be adjacent, for any $2 \le t \le n$.

For example, Figure I.1 illustrates a possible location of the travelers. In this example, traveler 1 is adjacent to travelers 2 and 10 but not adjacent to traveler 11.

11				
10	1	2		
9		3	4	
8	7	6	5	

Figure I.1: Example location of the travelers.

At each boarding step, all of the following happen simultaneously:

- The frontmost traveler in the queue, say traveler t, boards the aircraft and leaves the boarding area.
- For each t' ($t+1 \le t' \le n$), traveler t' takes the cell that traveler $t'-1$ was occupying immediately before the step.

For example, Figure I.2 illustrates the locations of the travelers after the first three boarding steps from the initial location above.

You are traveler p (that is, there are $p-1$ travelers in front of you). You know that your team coach is somewhere in the queue, but you do not know where. Assuming your team coach is equally likely to be any of travelers 1 to n (except p), you want to calculate the probability that you will be adjacent to your team coach at some point before you board the aircraft. Formally, you will be adjacent to traveler q at some point before you board the aircraft if there exists an integer s ($0 \le s < p$) such that traveler p and traveler q are adjacent after s boarding steps.

Input

The first line of input contains four integers r, c, n, and p ($1 \le r, c \le 1000; 2 \le n \le r \times c; 1 \le p \le n$). Each of the next r lines contains c integers. The j-th integer in the i-th line denotes $G_{i,j}$ ($0 \le G_{i,j} \le n$), where a non-zero

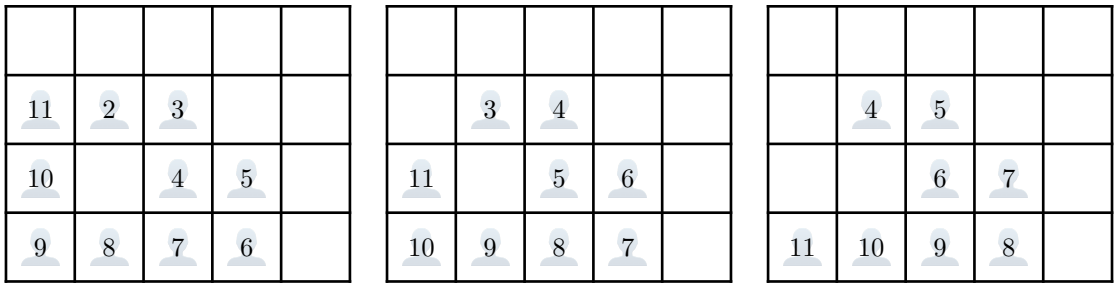


Figure I.2: Location of the travelers after 1, 2, and 3 boarding steps respectively.

value of $G_{i,j}$ means that traveler $G_{i,j}$ initially occupies the cell at row i and column j of the boarding area, while a zero value of $G_{i,j}$ means that no traveler occupies the cell. Across all pairs (i, j), each of the integers 1 to n appears exactly once in $G_{i,j}$. The input guarantees that traveler t and traveler $t-1$ are adjacent, for all $2 \le t \le n$.

Output

Output a fraction in the x/y format indicating the probability that you will be adjacent to your team coach at some point before you board the aircraft. The value of y must be equal to $n-1$. Note that there must not be spaces between the integers and the `/` delimiter.

Sample Input #1	Sample Output #1
4 5 11 2 11 0 0 0 0 10 1 2 0 0 9 0 3 4 0 8 7 6 5 0	3/10

Explanation for the sample input/output #1

This sample corresponds to the example given in the problem description above. You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 1, 3, or 11.

Sample Input #2	Sample Output #2
1 7 7 6 1 2 3 4 5 6 7	2/6

Explanation for the sample input/output #2

You will be adjacent to your team coach at some point before you board the aircraft if your team coach is either traveler 5 or 7. Note that the output `1/3` is **not** accepted.

Dress Rehearsal Problem J

Combination Lock

Time limit: 2 seconds

Your house is protected by a combination lock containing n rotating discs, numbered from 1 to n. On a typical combination lock, each rotating disc has 10 symbols, represented by integers between 0 to 9, inclusive. Since you are a mathematician, your combination lock is not typical. Instead, each rotating disc on your combination lock may have a different number of symbols. In particular, rotating disc i has $b_i - a_i + 1$ symbols, represented by integers between a_i to b_i, inclusive.

The combination lock is unlocked when each rotating disc displays one integer, and any pair of two integers displayed by the rotating disc are coprime. Two integers are coprime if they do not have any common positive factors other than 1.

You want to unlock the combination lock, so you want to determine what integer to be displayed on each combination lock to satisfy the requirement above. It is possible that your combination lock was sabotaged when you were gone so it might be impossible to unlock your combination lock.

Input

The first line of input contains one integer n ($2 \leq n \leq 50$). Each of the next n lines contains two integers. The i-th line contains a_i and b_i ($1 \leq a_i \leq b_i \leq 50$).

Output

Output one line containing n integers, where the i-th integer represents the integer to be displayed by rotating disc i to unlock the combination lock. If there are multiple solutions, you can output any of them. If there is no solution, output just the integer `-1`.

Sample Input #1	Sample Output #1
4 3 9 2 8 1 4 2 10	3 7 1 10

Explanation for the sample input/output #1

Other solutions such as `9 5 2 7` are also accepted.

Sample Input #2	Sample Output #2
3 5 6 5 6 2 6	-1

This page is intentionally left blank.

Dress Rehearsal Problem K
Candy Compress
Time limit: 4 seconds

You are developing a mobile game called *Candy Compress*. In this game, there are several colored candies lined up from left to right. There are 26 possible colors. At any point in time, the player can choose to add a candy at any position or to remove a subset of neighboring candies to get points depending on the colors of the removed candies.

To develop the game, you need to implement the following data structure. Initially, the data structure loads a 1-indexed string of n characters, which represents the colors of the initial candies. The string consists of only uppercase Latin characters (`A`–`Z`). After loading the string, there are q operations that you need to support. Each operation is either one of the following:

Operation 1: Insert the uppercase Latin character c to the string so that the character is in the i-th position in the new string. In particular, $i = 1$ means inserting character c at the beginning of the string. It is guaranteed that $1 \le i \le m + 1$, where m is the length of the string just before this operation.

Operation 2: Remove the characters of the string from the l-th to the r-th position, inclusive. It is guaranteed that $1 \le l \le r \le m$, where m is the length of the string just before this operation.

For each Operation 2, your data structure needs to determine the characters that are removed, so that the game can calculate the number of points to be given to the player. In other words, you need to determine the content of the string from the l-th position to the r-th position just before the operation.

Input

The first line of input contains two integers n and q ($1 \le n \le 500\,000$; $1 \le q \le 500\,000$). The second line contains a string consisting of n uppercase Latin characters representing the initial string to be loaded by the data structure. Each of the next q lines represents an operation with either one of the following formats:

1. `1` c i represents an Operation 1. It is guaranteed that c is an uppercase Latin character and $1 \le i \le m + 1$, where m is the length of the string just before this operation.
2. `2` l r represents an Operation 2. It is guaranteed that $1 \le l \le r \le m$ where m is the length of the string just before this operation.

The operations are given in the order they are to be performed. It is guaranteed that there is at least one Operation 2.

Output

For each Operation 2 in order, output one line containing the characters that are removed by the operation.

Sample Input #1	Sample Output #1
3 5 XPA 1 P 3 1 P 5 2 2 5 1 Y 2 2 1 2	PPAP XY

Explanation for the sample input/output #1

The first and second operations modify the string in the data structure as follows: `XPA` $\rightarrow$ `XP`**`P`**`A` $\rightarrow$ `XPPA`**`P`**. The third operation removes `PPAP` from the string, leaving the string `X` in the data structure. The fourth operation inserts the character `Y` to the end of the string. The last operation removes `XY` from the string, leaving an empty string in the data structure.

Sample Input #2	Sample Output #2
27 7 ICPCASIAPACIFICCHAMPIONSHIP 2 5 8 2 5 11 1 A 5 1 P 6 1 A 7 1 C 8 2 1 8	ASIA PACIFIC ICPCAPAC

Problem L
Secret Lilies and Roses
Time limit: 2 seconds

There are n flowers arranged in a line from left to right, which are numbered from 1 to n in that order. Each flower is either a lily or a rose. For an integer j between 0 and n, inclusive, let l_j denote the number of lilies among the leftmost j flowers, and let r_j denote the number of roses among the rightmost $n - j$ flowers.

Initially, only the number of flowers n is provided to you. The types of the flowers are hidden. You can obtain information on the flowers by making *queries*. In one query, you can perform one of the following.

Type query: Specify an integer i between 1 and n, inclusive. You will then receive the type of flower i.

Multiply query: Specify an integer j between 0 and n, inclusive. You will then receive the value of $l_j \times r_j$.

Your task is to find an integer k between 0 and n, inclusive, for which $l_k = r_k$ by making a limited number of queries. You can assume that at least one such integer exists for the arrangement of the flower types. Note that you do not need to identify the type of each flower.

Interaction

The first line of input contains one integer t ($1 \leq t \leq 100$) representing the number of test cases. After that, t test cases follow. Each of them is presented as follows.

The first line of each test case contains an integer n ($1 \leq n \leq 100$). After reading it, your program can start making queries. For each query, your program should write a line containing one of the following:

- "`type` i" to perform a Type query by specifying an integer i ($1 \leq i \leq n$). In response, an input line containing a string either `lily` or `rose` becomes available, indicating the type of flower i.
- "`multi` j" to perform a Multiply query by specifying an integer j ($0 \leq j \leq n$). In response, an input line containing the integer $l_j \times r_j$ becomes available.

Your program is allowed to make up to **10 queries** for each test case. This means the total number of Type queries and Multiply queries combined must not exceed 10. If your program makes more than 10 queries, it will be judged as "Wrong Answer".

When your program identifies an integer k such that $l_k = r_k$, it should write a line of the form "`answer` k" ($0 \leq k \leq n$). Note that providing the answer does not count as a query.

The input guarantees that at least one correct output exists. If there are multiple correct outputs, any one of them will be accepted.

After writing the answer line, your program should start processing the next test case. When all t test cases have been processed, the interaction stops and your program should terminate.

If your program is judged as "Wrong Answer", an input line containing -1 becomes available. After that, the interaction stops without processing the remaining test cases, and your program should terminate.

Notes on interactive judging:

- The evaluation is non-adversarial, meaning that the types of the flowers are chosen in advance rather than in response to your queries.
- Do not forget to flush output buffers after writing. See the "Judging Details" document for details.
- You are provided with a command-line tool for local testing, together with input files corresponding to the sample interactions. You can download these files from DOMjudge. The tool has comments at the top to explain its use.

Read	Sample Interaction #1	Write
2		
9		
		type 8
lily		
		type 1
rose		
		multi 6
6		
		multi 3
3		
		answer 5
3		
		multi 3
0		
		type 3
rose		
		answer 3

Explanation for the sample interaction #1

For the first test case, nine flowers are arranged as shown in the following figure. In response to the third query, $l_6 \times r_6 = 3 \times 2 = 6$ is returned, and in response to the fourth query, $l_3 \times r_3 = 1 \times 3 = 3$ is returned. Since $l_5 = r_5 = 3$, $k = 5$ is a correct output.