Problem A. Guess the String

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MiB
Queries limit:	75

This is an interactive problem.

A non-empty string S consisting of not more than 10 lowercase English letters is fixed but not revealed to you. Your task is to find that string by making queries. Each query is a mask consisting of lowercase English letters and "`*`" characters. The character "`*`" matches any (possibly empty) sequence of lowercase English letters.

Interaction protocol

After each query, jury program answers if that mask matches the string S. If the answer is positive for a query that contains no "`*`" characters, the goal is achieved.

It is guaranteed that the provided number of queries is enough to find any possible string S which satisfies the constraints.

Output

Each query must contain a non-empty string of no more than 21 lowercase English letters or "`*`" characters (ASCII 42).

The number of queries must not exceed 75.

Don't forget to `flush` the standard output after printing each line.

Input

The answer for each query is given on a separate line. Line with answer contains "`Yes`" if the mask matches string S, "`No`" if not, and "`Exit`" if the program must be terminated.

Example

standard output	standard input
`*aca*`	`Yes`
`*a*a*a*a*`	`Yes`
`aca*`	`No`
`*aca`	`No`
`aba`	`No`
`aba*aba`	`Yes`
`abacaba`	`Exit`

Problem B. Mine Field

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MiB
Queries limit:	10 000

This is an interactive problem.

You are on a rectangular cellular field of size $N \times M$ ($2 \le N, M \le 10$). The field is toroidal. This means you can mentally join left and right borders of the field, as well as top with bottom.

Some cells contain mines, and one of the empty cells contains a treasure. You cannot move onto mines. Your goal is to find the treasure. You don't know field size as well as your position on the field. It is guaranteed that the starting cell contains no mine and no treasure. You can move to any of the four cells sharing a side with your cell, or perform a cyclic shift of your row or column in one of the possible directions. You can shift only the column or row where you are. During a shift, everything in the respective row or column is shifted including yourself, mines and treasure.

The treasure is considered found if you moved in such a way that for every possible starting position of the treasure, you arrived at the same position as the treasure at some moment of time. It is guaranteed that it is possible to find the treasure.

Interaction protocol

At the start and after each move or cyclic shift, jury program either gives you the total number of mines in row and column that contain the participant's cell, or asks you to exit the program (if the player already found the treasure).

Output

Each line must contain one character which must be one of the following: "`l`", "`u`", "`r`" and "`d`" for moving left, up, right or down, respectively; or "`L`", "`U`", "`R`" or "`D`" for performing cyclic shifts to the left, upwards, to the right or downwards, respectively.

The total number of commands must not exceed 10 000.

Don't forget to `flush` the standard output after printing each line.

Input

Each line contains one integer — the total number of mines or -1 if you must exit the program.

Example

standard output	standard input
	5
R	3
d	3
d	3
L	5
L	4
U	3
d	-1

The following field is used in the example:

Problem C. Resistance

Input file:	`standard input`
Output file:	`standard output`
Time limit:	12 seconds (15 seconds for Java)
Memory limit:	256 MiB

This is an interactive problem.

In this problem, we will consider a simplified version of the popular game "Resistance". N players participate in the game. At the start of the game, cards with roles are given to players in random order. Each player does not know roles of other players until the end of the game. There are two roles: a Resistance activist and an Empire spy. The goal of each side is to get three score points. Points are acquired by completing or sabotaging missions.

Each game consists of 3–5 missions. In this problem, you decide which players will take part in each mission. You don't decide anything more. Each player taking part in a mission decides whether he will carry out or sabotage the mission (it is not known what they decided). Resistance activists always carry out the mission. If no one sabotaged a mission, one point goes to Resistance activists. Otherwise, one point goes to the Empire.

Here is the distribution of roles depending on the number of participants:

Players number:	5	6	7	8
Resistance activists number:	3	4	4	5
Empire spies number:	2	2	3	3

Here is the number of players required for each mission:

The number of players:	5	6	7	8
Mission 1:	2	2	2	3
Mission 2:	3	3	3	4
Mission 3:	2	4	3	4*
Mission 4:	3	3	4*	5*
Mission 5:	3	4	4	5

(*) at least two sabotage required for Empire spies to get a point.

Missions are always performed in the order above.

The strategy of spies is the following. At the start of the game, each spy chooses a random real number from 0.7 up to 1 equiprobably and independently from others. This number is the chance for him to sabotage each mission. If the Empire already got two points, each spy will sabotage each mission.

Interaction protocol

Each test consists of 4 000 games. In each game, jury program starts by giving you the number N. Then follow the descriptions of rounds. At the start of each round, jury program asks you to choose the players which will take part in the mission. Participant program must output the numbers of chosen players and get the number of sabotages or game end signature in return.

Solutions for this problem will be checked on four tests:

1. $n = 5$ in all games.
2. $n = 6$ in all games.
3. $n = 7$ in all games.
4. $n = 8$ in all games.

A solution will be considered right if Resistance won at least 49% of games.

Input

Before the first round of game "Resistance", jury program outputs the positive integer N — the number of players.

Round format:

At the start of each round, jury program outputs one positive integer K, the number of players which must take part in the mission.

After the participant outputs the list of players, jury program outputs one line which contains either the number of sabotages or the text "`WIN`" or "`LOSE`" if Resistance activists won or lost, respectively.

The last line contains the only number $N = -1$. You shold not process this game and must just exit the program.

Output

For each request for a list of players to take part in a mission, participant program must output a list of numbers of players $A_1, A_2, \ldots, A_K$ ($1 \le A_i \le N$, $A_i \ne A_j$ for $i \ne j$) which will take part in that mission.

Don't forget to `flush` the standard output after printing each line.

Example

standard input	standard output
5 2 1 3 1 2 LOSE 5 2 1 3 0 2 1 3 0 3 WIN -1	 3 5 1 2 3 1 4 1 2 1 4 5 4 5 1 3 5 1 3 5

This example does not correspond to the first test. The first test consists of 4 000 games.

In the first game of this example, spies are players with numbers 3 and 4, and in the second one, spies have the numbers 2 and 4.

Problem D. Robot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MiB

This is an interactive problem.

You are given a square field of size 81×81. The coordinates of the cells are non-negative integers from 0 to 80. There is a robot in the center of the field (cell with coordnates $(40, 40)$). The robot and the player make moves in turn. First, the robot moves on the field in one of the four directions. After that, the player burns one of the cells of the field. The robot's goal is to leave the field by moving beyond its borders. The player's goal is to prevent that from happening.

Robot can move according to the special rules: if current move direction is equal to previous move direction, it can move in this direction by distance from 1 up to $x + 1$ where x is the length of previous move. If movement direction changed, he can move by only one cell (distance equal to one). Robot can not fly or teleport, he moves around the field, so he can not go through burned cells.

The player can not burn a cell twice, as well as burn the cell containing the robot.

Interaction protocol

Jury and participant programs make moves in turn. As the robot moves first, jury program moves first. It outputs either "`E`" which means that the robot surrenders and the player wins, or the direction and length of robot move, then waits for participant turn. Participant program must output coordinates of a cell to burn.

Input

Jury program can output commands "`L` x", "`R` x", "`U` x", "`D` x", "`E`". If jury program outputs "`E`", it means that the robot surrenders and you must exit your program. Other commands mean robot movement: direction (left, right, up or down) and move length; in this case, jury program waits for the player's next turn. The upper left corner has coordinates $(0, 0)$, and the lower right corner has coordinates $(80, 80)$. Moving to the left decreases the column number. The upward movement, in turn, decreases the row number.

Output

After each move of the robot, participant program must output two integers in range from 0 up to 80: row number and column number of a cell to burn.

Don't forget to `flush` the standard output after printing each line.

Example

standard input	standard output
L 1 U 1 U 2 D 1 D 1 D 1 E	40 38 36 39 37 38 41 39 40 40 39 39

Problem E. Elevators

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MiB
Queries limit:	10,000

This is an interactive problem.

There are N ($2 \le N \le 100$) floors and K ($1 \le K \le 6$) elevators in the building. You are standing on the ground floor of the building and you need to get to the floor where an important event is held. You have no idea where the event is held exactly, so be ready to examine every floor.

There is a little problem with these elevators — they have only two buttons inside: up and down. When you press one of these buttons, elevator moves by A_i ($1 \le A_i < N$) floors towards corresponding direction if possible and does nothing otherwise. You know that each elevator has its own constant A_i, but again, you don't know exact values. Furthermore, the floors are indistinguishable except your destination floor.

It is guaranteed that every floor is accessible through the elevators.

Interaction protocol

In the very beginning you are given two numbers N and K. Then for each command you'll get one of three responses: elevator successfully moved, elevator ignored your command, or you visited all floors. You shouldn't issue any commands after you have visited all floors.

Output

Each line of the output should contain either positive of negative integer x ($1 \le |x| \le K$). Positive number means that you have entered the elevator number x and pressed the "up" button. Negative number means the same except that you pressed the "down" button.

The number of lines shouldn't exceed 10 000.

Don't forget to `flush` the standard output after printing each line.

Input

First line contains two numbers N and K. Other lines contain either "`Yes`" if command has been successfully executed, "`No`" if command has been ignored, or "`Quit`" if you visited all floors.

Example

standard output	standard input
	6 2
1	Yes
1	Yes
1	No
-2	Yes
-2	No
-1	No
1	Yes
1	Quit

Values for this example: $N = 6$, $K = 2$, $A_1 = 2$, $A_2 = 3$.

Problem F. Maze

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MiB
Queries limit:	100 000

This is an interactive problem.

The maze is a rectangular grid. Its dimensions don't exceed 30. Each cell of the grid can be either empty, or be an obstacle, or contain a wormhole which is one of two ends of bidirectional teleportation tunnel. Each wormhole corresponds to exactly one tunnel. The number of tunnels doesn't exceed 10.

You are inside the maze, holding a single stone in your hand (this is the only stone in the whole maze). You are acting stepwise. During a single step, you can only try to move one cell up, down, left or right; or try to drop the stone in the current cell; or try to pick the stone up. When you move from a neighboring cell to the wormhole, you are teleported to the other end of the tunnel and stay there. If you are standing on a cell with a wormhole in the beginning of your step, that wormhole will neither interfere with you, nor with your dropped stone.

The problem is to determine the number of reachable cells.

It is guaranteed that you cannot walk outside the maze, there are no neighboring wormholes and every wormhole has at least one empty neighboring cell. Also, it is guaranteed that the starting cell contains no wormhole.

Interaction protocol

As soon as you are ready to say the number of reachable cells, you should output the answer with a special command and terminate your program. For all other commands you'll get one of three responses: the cell you tried to move to contains an obstacle (hence you didn't move), the cell you are standing at the end of the step is empty, or the cell you are standing at the end of the step contains your dropped stone.

Output

Each line should contain a single command:

- `MOVE dir`: try to move towards `dir`. `dir` can be either `UP` or `DOWN` or `LEFT` or `RIGHT`.
- `DROP`: try to drop the stone.
- `TAKE`: try to pick the stone.
- `DONE x`: output the answer `x` — the number of reachable cells.

If you issue the `DROP` command but don't have the stone, or issue the `TAKE` command but the stone doesn't lie in your cell, the stone does not move, and the player just gets the type of his cell.

The total number of commands shouldn't exceed 100 000.

Don't forget to `flush` the standard output after printing each line.

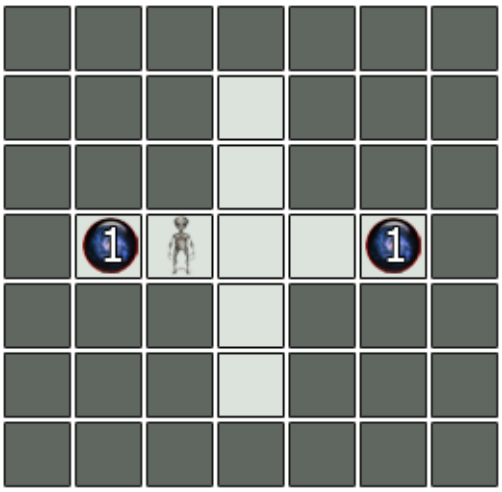
Input

Each line contains either "`WALL`", if you are trying to hit an obstacle, "`EMPTY CELL`" if you are on the empty cell, or "`CELL WITH STONE`" if you are on the cell with your dropped stone.

Example

standard output	standard input
MOVE RIGHT	EMPTY CELL
DROP	CELL WITH STONE
MOVE RIGHT	EMPTY CELL
MOVE RIGHT	EMPTY CELL
MOVE RIGHT	EMPTY CELL
MOVE RIGHT	CELL WITH STONE
MOVE UP	EMPTY CELL
MOVE UP	EMPTY CELL
MOVE UP	WALL
MOVE DOWN	EMPTY CELL
MOVE DOWN	CELL WITH STONE
TAKE	EMPTY CELL
MOVE DOWN	EMPTY CELL
MOVE DOWN	EMPTY CELL
MOVE DOWN	WALL
DONE 9	

The maze from the example:



Problem G. Reflections

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MB
Queries limit:	2 000

This is an interactive problem.

Consider N-dimensional linear space $\mathbb{R}^N$, where distance between points x and y is determined as follows. Let apply to both points linear mapping A, whose norm doesn't exceed 100, resulting in images $\tilde{x}$, $\tilde{y}$. So, distance between x and y equals to:

$$\sqrt{\sum_{i=1}^{N}(\tilde{x}_i - \tilde{y}_i)^2}.$$

You are drawing a polyline with a pen. Initially pen is placed at zero point (origin of the coordinate system). There are $2 \cdot N$ hyperplanes, each of which can be used as a symmetry axis to reflect the position of the pen. Also you can draw the next segment to know out its length.

The problem is to calculate the distance between given points a and b.

Interaction protocol

Initially you are given a number N, description of hyperplanes and coordinates of a and b points.

For each "reflect" command you'll get reflected position of the pen. For each "draw" command you'll get the length of drawn segment. For each answer output you'll get a judgement about correctness of your answer.

You should terminate your program after correct answer only. The absolute or relative error of 10^{-3} is allowed.

Output

Each line should contain the only command:

- `Reflect` x — reflect the pen over hyperplane x.
- `Request` — draw the segment of the polyline.
- `Answer` d — output the answer d — distance between a and b points.

The number of commands shouldn't exceed 2 000.

Don't forget to `flush` the standard output after printing each line.

Input

First line contains single integer N ($2 \leqslant N \leqslant 10$) — space dimension. It's folowed by description of $2 \cdot N$ hyperplanes: for each hyperplane there is line containing coordinates of hyperplane's normal followed by line containing coordinates of some point that belongs to hyperplane.

All numbers are integer. All normal's coordinates does not exceed 100 by it's absolute value, coordinates of points does not exceed 1 000 by it's absolute value. Normal of hyperplanes numbered from 1 up to N are linearly independent; the same is true for hyperplanes numbered from $N + 1$ up to $2 \cdot N$. No two hyperplanes are parallel. There is no hyperplain that contains origin of coordinate system.

Each of the following two lines contains N integers, that does not exceed 10 000 by it's absolute value — coordinates of points a and b respectively.

Each of the following lines contains answer to some request:

- `Reflect`: N real numbers with 9 signs after the decimal point — pen's coordinates.
- `Request`: single real numberc with 9 signs after the decimal point — segment's length.
- `Answer`: single symbol: «N» in case answer is incorrect, and «Y» otherwise.

Example

standard output	standard input
	2 1 0 2 0 -1 1 0 -1 0 1 0 -2 1 1 2 2 4 2 10 4
Answer 1	N
Answer 1.5	N
Reflect 1	4.000000000 0.000000000
Reflect 2	1.000000000 3.000000000
Reflect 1	3.000000000 3.000000000
Reflect 2	4.000000000 2.000000000
Request	8.944271910
Reflect 3	4.000000000 -6.000000000
Reflect 1	0.000000000 -6.000000000
Reflect 4	10.000000000 4.000000000
Request	12.649110641
Answer 12.64	Y

In the given sample mapping doubles the coordinate values.

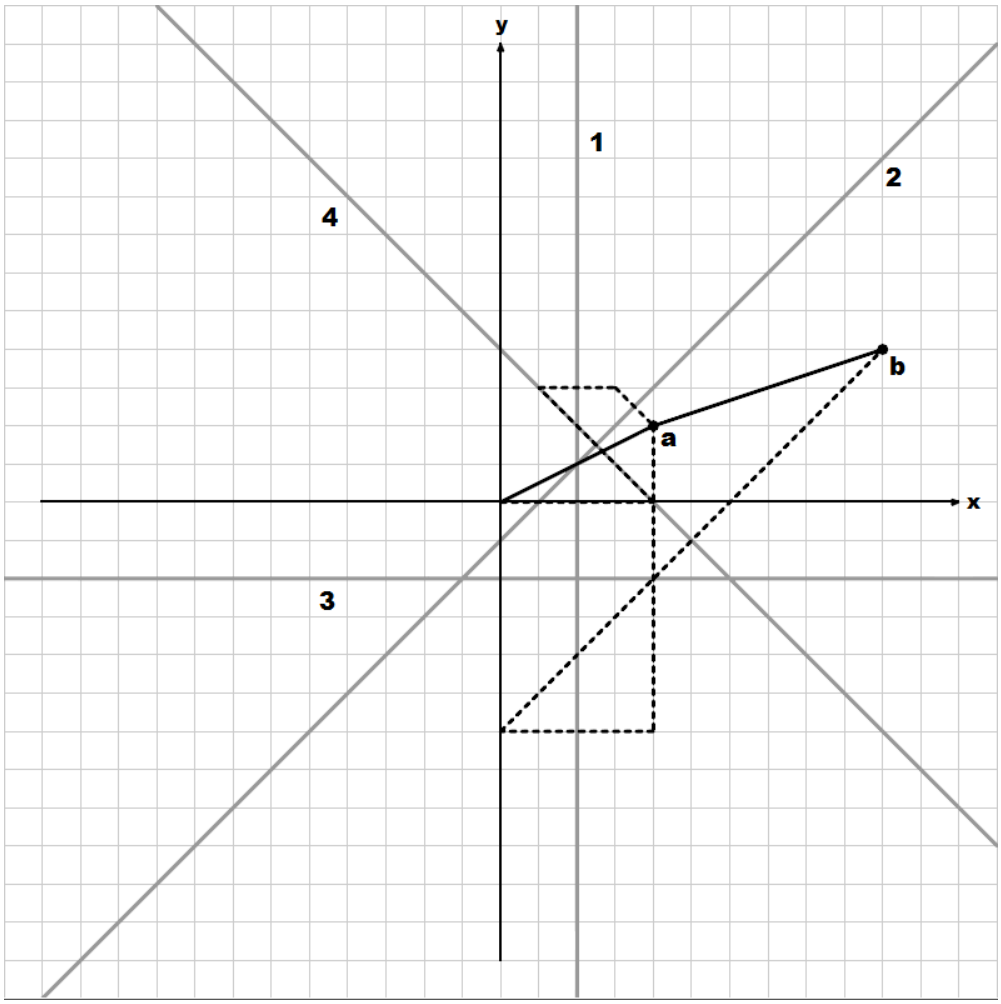
Note

Linear mapping $A : \mathbb{R}^N \to \mathbb{R}^N$ is mapping that satisfies to the linearity properties: $A(x+y) = A(x)+A(y)$ и $A(\alpha x) = \alpha A(x)$ for each $x, y \in \mathbb{R}^N$ и $\alpha \in \mathbb{R}$.

Norm of the linear mapping is supremum of value $||A(x)||$ for each point x that satisfies $||x|| = 1$. Here $||x||$ is the usual euclidian distance from origin to x.

The set of vectors v_1, v_2, ..., v_k is called Linearity independent if for any set α_1, α_2, ..., α_k consisting of k numbers, at least one of which is not equal to zero, the vector $\alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k$ is not equal to zero vector.

The following image corresponds to the sample:



Problem H. Shooting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 MB

This is an interactive problem.

Someone standing on the earth throws the ball having radius 10 cm in some direction and with some angle (not less than 45 degrees) from the ground. The velocity of the ball is 100 m/sec. There is a shooter who wants to hit the ball. He stands at some point on the earth. The distance between the shooter and the centre of the ball is 1 050 meters. This distance is quite large so the shooter can not see how the ball moves. The shooter never moves.

In this problem problem we consider that the earth is flat, the shooter is a material point, and the acceleration of gravity is 9.80665 m/sec^2.

For convenience the shooter mentally draws a rectangular coordinate target in the following way. Target is located between the shooter and the ball in the plane that is perpendicular to the line connecting the position of the shooter and the centre of the ball (at the starting moment of time). This plane is located at the distance of 50 m from the shooter. The target is 1000 m high and 2000 m wide. This target is located on the ground and its lower edge is symmetric relative to the line connecting the shooter and the centre of the ball. There is a Cartesian coordinate system introduced on this target as follows: the upper left corner (when viewed from the shooter) has coordinates $(-1000, 1000)$, and the lower right one — $(1000, 0)$.

The shooter shoots in the following way: he chooses some point on his target and shoots in the direction of this point. The shot is a ray starting at the position of the shooter and passing through the chosen point. After the shot the shooter on the radio knows the distance between the shot and the centre of the ball. At the moment of time 0 the ball is lying on the ground, and the shooter is on the same height as the centre of the ball.

The first shot is made 0.5 seconds after the throw. Each of the following shots is made 0.5 seconds after the previous one.

Now then, you are the shooter and your target is to hit the ball before he reached the earth.

Interaction protocol

After each shot the jury program responds the distance between the ray and the centre of the ball or requests the termination of the program.

Output

Each line of output should contain two real numbers x and y ($-1000 \leqslant x \leqslant 1000$, $0 \leqslant y \leqslant 1000$) — the coordinates of the point on the target.

Don't forget to `flush` the standard output after printing each line.

Input

Each line of input contains «`Miss` d», is the shooter missed the ball, where d is the distance between the ray and the centre of the ball (in meters), or «`Quit`», if the program should be terminated.

Example

standard output	standard input
0 0 0 1 0 3 0 6.7	Miss 34.130339059 Miss 46.215649475 Miss 38.336035338 Quit

In the given sample the throw is directed to shooter and the throw angle is 45 degrees.

Problem I. Conveyor

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MiB
Queries limit:	10 000 `Next` commands, 10 successive `Swap` commands

This is an interactive problem.

There is a cyclic conveyor with N ($20 \le N \le 100$) balls. Some of them are black and the others are white. Through your working window, you can see only 5 adjacent balls on the conveyor at a time. You are permitted to swap any two visible balls or shift the conveyor in counter-clockwise direction by S ($1 \le \mathrm{GCD}(N, S) \le 5$) balls. Here, $\mathrm{GCD}(x, y)$ is the greatest common divisor of integers x and y. Your task is to determine the number of black balls.

The initial state doesn't contain 5 or more black balls in succession. The number N is not given.

Interaction protocol

The jury program outputs colors of 5 visible balls in clockwise order at the start and after each user command. If the participant found the answer, he can print it and exit the program immediately after that.

Output

Every string should be either "`Next`" to shift the conveyor, "`Swap` a b" to swap visible balls with numbers a and b (visible balls are numbered from 1 clockwise), or "Answer x" to output the answer.

The total number of "`Next`" commands should not be greater than 10 000. There should be no more than 10 successive "`Swap`" commands.

Don't forget to `flush` the standard output after printing each line.

Input

Every string has 5 characters each of which is either "`W`" for a white ball or "`B`" for a black ball. Visible balls are listed in clockwise direction.

Example

standard output	standard input
	BWWWW
Swap 1 3	WWBWW
Swap 3 5	WWWWB
Next	WBWWW
Swap 2 5	WWWWB
Next	WWBWW
Swap 3 5	WWWWB
Next	WWWBW
Swap 1 4	BWWWW
Next	WWWWW
Next	BWWWW
Swap 1 5	WWWWB
Next	BWWWW
Swap 1 5	WWWWB
Next	BBWWW
Next	WWWWW
Next	WWWWW
Next	BWWWW
Swap 1 5	WWWWB
Next	BBBWW
Next	WWWWW
Next	WWWWW
Next	WWWWW
Next	BBBBW
Answer 4	

In this example, $N = 21$ and $S = 5$.

The initial position is the following:

Problem J. Questionnaire

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds (4 seconds for Java)
Memory limit:	256 MB
Queries limit:	15

This is an interactive problem.

You need to give correct answers to all 18 questions of a questionnaire. There are 4 possible answers for each question: "a", "b", "c", and "d".

Correct answers for every two consequent questions can't be the same.

You can make several attempts to answer all questions. After each attempt, you will get the total number of correct answers on this attempt.

Interaction protocol

For every set of answers to the questionnaire, jury program will output the total number of correct answers, or an instruction to terminate the program.

Output

Each attempt is a separate string of 18 characters each of which is either "a", "b", "c" or "d". Character number i represents the answer for i-th question.

The total number of attempts must not be greater than 15.

Don't forget to `flush` the standard output after printing each line.

Input

After each attampt, you get one number on a line: the number of correct answers, or -1 if you should terminate the program.

Example

standard output	standard input
aaaaaabbbbbbcccccc	5
bbbbbbddddddaaaaaa	7
abababababababab	6
abacababababaabacaba	14
abacabaddddabacaba	16
abacabadadaabacaba	15
abacabadbbdabacaba	17
abacabadbcdabacaba	-1