Problem A. Two Instructions

Input file:	`2instr.in`
Output file:	`2instr.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Probably, the most important instruction in 8086 is "`pop sp`" which takes the stack pointer value from the stack.

In addition, in 80486 processor Intel has added a second instruction, also very important for working with stack pointer register — "`bswap sp`". It swaps the high and the low bytes of this register.

- "`bswap sp`" has the instruction code `0F CC`;
- "`pop sp`" has the instruction code `5C`;
- memory segment is 65536 bytes long;
- memory is considered looped, that is after the cell FFFF goes cell 0000;
- "`pop sp`" takes a big-endian 16-bit word from the memory pointed by `sp` and loads it into `sp`;

You are given some initial `sp` value and the contents of the memory. Find the shortest possible program such that after its execution the value of stack pointer is equal to another given value.

Input

The first line of the input file contains two hexadecimal 16-bit numbers from 0000 to FFFF — the starting and ending values of `sp` register. The next 4096 lines contain hex dump of the segment.

Hex dump is displayed in format [8 bytes][2 spaces][8 bytes] where [8 bytes] are ([byte][space]*7)[byte]; [byte] is a hexadecimal number from 00 to FF with exactly two digits.

Output

In the first line output the number of code bytes in the shortest program (in decimal notation). Then you should output the hex dump of the resulting program in the same format as in input. The last (possibly incomplete) line of hexdump must be immediately terminated by the end-of-line character after the last byte output.

If there are several possible shortest programs, output the lexicographically smallest one.

If it is impossible to solve task for a given dump and values, write a single line containing the word `IMPOSSIBLE`.

Example

2instr.in
0003 00EF 00 00 EF 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 *above line repeats 4094 times more*
2instr.out
4 5C 0F CC 5C

Problem B. The Strange Computer

Input file:	`japan.in`
Output file:	`japan.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given the strange computer which initially contains in memory only one integer x. Your task is to do so that its memory will contain numbers $a_1x, a_2x, \ldots, a_nx$ (and may be some others) with the minimal possible number of operations. The operation is one of the following:

1. Add any two numbers from memory
2. Subtract any two number from memory
3. Binary shift left of k bits (i.e., take any value from memory and multiply it by 2^k)

Results of all the operations are stored in memory and can be used for further operations.

The values more than $42x$ and less than zero are not allowed.

The order in which $a_1x, a_2x, \ldots, a_nx$ appear in memory does not matter.

Input

Ther first line contains n ($1 \leqslant n \leqslant 41$). The second line contains n numbers a_i ($2 \leqslant a_i \leqslant 42$). a_i are different. x is not given, the sequence of operations should lead to result independently of x.

Output

At the first line output the minimal number of operations. The following lines contain operations, one by line, in the following format:

1. Shift left ax by k bits: "`a<<k`"
2. Add ax and bx: "`a+b`"
3. Subtract ax from bx: "`b-a`"

No spaces are allowed in output.

Examples

japan.in	japan.out
3 3 5 18	5 1<<1 1<<4 1+2 2+3 2+16
1 29	4 1<<1 1<<5 1+2 32-3
4 12 19 41 42	8 1<<1 1<<4 1+2 3<<2 3+16 19<<1 3+38 1+41

Problem C. Simple Game

Input file:	`game.in`
Output file:	`game.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two players play a simple game on a $n \times n$ board. The first player has a single white checker and the second player has a single black one. The two players alternate turns, and the first player moves first.

When it is the first player's turn, he chooses one of four directions (up, down, left or right) and moves his checker one cell in the chosen direction. When it is the second player's turn, he also chooses one

of those four directions and moves his checker one or two cells in the chosen direction. A player wins the game when his move puts his checker in the cell occupied by his opponent's checker.

Both players use an optimal game strategy. If the player can win, he will follow the strategy that minimizes the number of moves in the game. If the player cannot win, he will follow the strategy that maximizes the number of moves in the game.

Input

The input file consists of five numbers n ($2 \leqslant n \leqslant 20$), and the coordinates of white and black checker. These two cells are different.

Output

Output `WHITE` x, if white wins, `BLACK` x, if black does, `DRAW`, if the game will end in a draw. Here x is the number of moves of both sides until the game ends.

Example

game.in	game.out
2 1 1 2 2	BLACK 2
2 2 2 1 2	WHITE 1
3 1 1 3 3	BLACK 6

Problem D. Graph Game

Input file:	gg.in
Output file:	gg.out
Time limit:	2 seconds
Memory limit:	256 megabytes

> Two trolls play an interesting game: they break each other's legs. The one who can't move, loses.
>
> Troll's game theory

You are given a directed graph of some game for two players and the initial position in this game. Notice that in a graph game each player is allowed to make move from some position to every position, that has an edge coming to it from the current one. Player take alternating moves; the one who cannot make move, loses. You have to check whether the first player wins regardless of players' moves.

Input

Input contains one or more test cases. The first line of each test case contains the number of vertices V, the number of edges E ($1 \leqslant V \leqslant 100\,000$, $1 \leqslant E \leqslant 100\,000$) and the number of starting position a ($1 \leqslant a \leqslant V$). E lines follow, containing descriptions of edges in format: u_i v_i ($1 \leqslant u_i, v_i \leqslant V$), meaning edge from vertex u_i to the vertex v_i. The input is terminated with three zeroes. Sum of all E in the input doesn't exceed 100 000, the number of test cases doesn't exceed 1000.

Output

Follow the output sample as precisely as possible — the judgement is carried out automatically.

Example

gg.in
3 2 1 1 2 1 3 1 1 1 1 1 0 0 0
gg.out
First player always wins in game 1. Players can avoid first player winning in game 2.

Problem E. Game with the Keyboard

Input file:	keyboard.in
Output file:	keyboard.out
Time limit:	7 seconds
Memory limit:	256 mebibytes

Masha and Misha play the following game. At first, each of them writes some words on the piece of paper. Then they take a keyboard and do in turn l moves — each move consists of removing one of the English letters from the keyboard.

At the end, let A is a number of Masha's words she can type with the remaining letters, and B is the same for Misha. So Masha's score is $A - B$ and Misha's score is $B - A$.

Masha moves first. Your task is to find how much score she can get assuming both players play optimally, maximizing their own score.

Input

The first line contains the number of turns l ($1 \leqslant l \leqslant 26$). Masha's words follow, the next line and the number of Masha's words and the next one is the list of words separated by spaces. Then Misha's word are written in the same format.

All words are of small English letters, each player wrote no more than 15 words each consisting of no more than 30 letters.

Output

Output optimal Masha's score.

Example

keyboard.in	keyboard.out
2 5 abacaba a zxxzyz trava abc 1 a	1

Problem F. The King and the Rook

Input file:	kingrook.in
Output file:	kingrook.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given a small chessboard 6×6. There are three pieces on it: a White King, a White Rook and a Black King. Your task is to calculate the minimal number of moves required for white side to checkmate the Black King or determine that it is impossible or that the position is incorrect.

Input

The only line of the input file contains three coordinates of fields — coordinates of White King, White Rook and Black King respec-

tively, followed by an identifier of the side to move (`W` for white or `B` for black).

Output

Write to the output file total number of moves (for both sides) required for white to win the game. If the black king is already checkmated, output 0. If the input position is incorrect (two kings are adjacent or it is white move and the check is declared for black), output -1. If the game will end in a draw (e.g. black are stalemated), output -2. Both players play optimally.

Example

kingrook.in	kingrook.out
c6 f4 a5 B	2
c6 f4 b5 W	-1

Problem G. Strange Two-player Game

Input file:	`labyrinth.in`
Output file:	`labyrinth.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

The game on arbitrary directed graph is played as follows: at the beginning there is a pawn in one of the graph vertices (this vertex is called a starting vertex). Two players move in turn, one move is to select an edge from the current vertex and move the pawn to the other end of it. The player who can't move loses.

Instead of playing optimally, two players are doing something else. If the player who moves has a winning strategy regardless of her opponent moves, she will try to play as long as possible, provided that she is still winning, to get as much fun of being a winner as possible. If the player who moves loses regardless of doing anything (her opponent has a winning strategy), she will try to end the game as fast as possible, because knowledge of being a loser is very bad.

Your task is to determine the winner, and the number of moves the game will last if both sides use such strategy.

Input

First line contains two integers n and m "— the number of vertices and edges ($1 \leqslant n \leqslant 100\,000$, $0 \leqslant m \leqslant 100\,000$). m lines follow, each describing an edge by two integers a and b — the edge goes from a to b.

Output

Output n lines — for each vertex the result of the game starting from this vertex. If no player has a winning strategy, output «`DRAW`». If the starting player has a winning strategy, output «`WIN K`», where `K` is the number of moves the game will last. If the starting player has a winning strategy and can do so that the game lasts forever (and she still has a winning strategy), output «`WIN INF`». If the starting player's opponent has a winning strategy, output «`LOSE K`», where `K` is the number of moves the game will last. But if she loses and her opponent can make the game to last forever, output «`LOSE INF`».

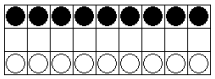
Examples

labyrinth.in	labyrinth.out
4 4 1 2 1 3 2 4 3 4	LOSE 2 WIN 1 WIN 1 LOSE 0
6 6 1 2 2 3 3 4 4 1 4 5 5 6	DRAW DRAW DRAW DRAW WIN 1 LOSE 0
6 6 1 2 2 3 3 4 4 1 2 6 4 5	LOSE INF WIN INF LOSE INF WIN INF LOSE 0 LOSE 0

Problem H. Game of Pawns

Input file:	`pawns.in`
Output file:	`pawns.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two players play in game with pawns on a board $3 \times N$:



The pawns are usual chess pawns and move and capture as pawns in chess, so, they can only move one cell forward if its empty (white moves up, black moves down) and capture the pawn of the other color one cell diagonally forward (the captured pawn is removed from the board). But if you can capture the other color, it is required to do so in any possible way. White moves first. The player who can't move, loses. Your task is to determine the winner assuming both players play optimally.

Input

An integer N ($1 \leqslant N \leqslant 10^9$).

Output

«White» or «Black» depending on the winner.

Example

pawns.in	pawns.out
3	White
4	Black
5	White