PEBBLES (100 points)

Alice and Bob are playing a board game with some pebbles. The board is set up as a strip divided into N squares, numbered 0 to N-1.

There are a few variations of this game:

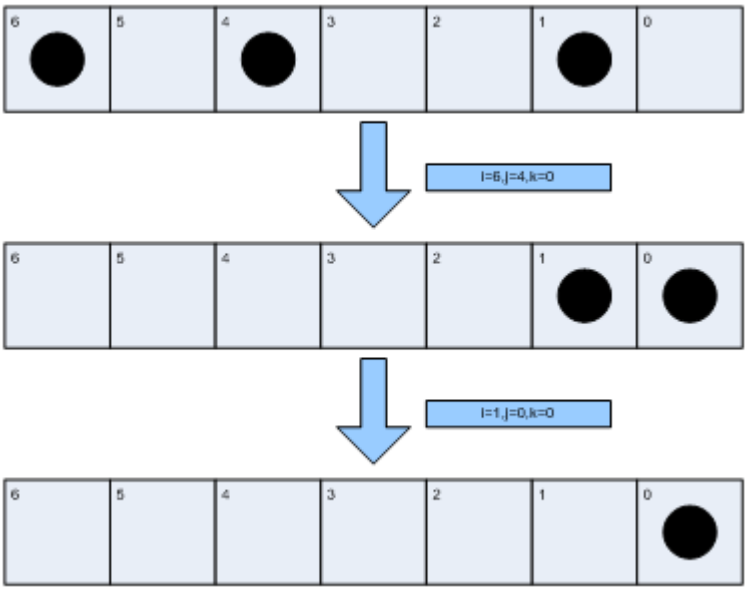
Part 1 (30 points)

In this variation, N<=20 and each square on the strip can be empty, or it can contain a single pebble. A move in this game is performed in the following manner:

- Choose three numbers *i*, *j* and *k*, such that 0<=*i*,*j*,*k*<N, *i*>*j*, *i*>*k*, and there is a pebble in square *i*. (Note that it is acceptable if *j*=*k*).
- Remove the pebble from square *i*.
- If there is a pebble in square *j*, remove it. Otherwise, add a pebble to square *j*.
- If there is a pebble in square *k*, remove it. Otherwise, add a pebble to square *k*.

Alice and Bob take turns moving, with Alice making the first move. Each player must make a single legal move during his/her turn (nobody can pass). A player loses if there are no legal moves available to them, at which point the game ends.

Here is an example game with N = 7:



Alice begins by choosing *i*=6,*j*=4,*k*=0. The pebble in square 6 is removed, the pebble in square 4 is removed and a pebble is added in square 0.

Next, Bob chooses *i*=1,*j*=*k*=0 (the only move available). The pebble on square 1 is removed; the pebble on square 0 is removed but then added back again.

No more legal moves are available, so Bob wins.

Part 2 (40 points)

In this variation, N<=20 and each square on the strip can contain any number of pebbles. A move in this game is performed in the following manner:

- Choose three numbers *i*, *j* and *k*, such that 0<=*i*,*j*,*k*<N, *i*>*j*, *i*>*k*, and there is at least one pebble in square *i*. (Note that it is acceptable if *j*=*k*).

- Remove one pebble from square *i*.
- Add one pebble to square *j*.
- Add one pebble to square *k*.

Part 3 (30 points)
This is the same as part 2, with the exception that N<=100.

TASK
Write a program that, given an initial game state, determines the winning strategy, if one exists. This means that assuming each player always makes the best possible move, find out if the player whose turn it is to move, can move in a manner which will guarantee a win.

CONSTRAINTS
1 <= N <= 20 for parts 1 and 2
1 <= N <= 100 for part 3
In the initial game state, the number of pebbles in a square does not exceed 1 for Part 1 and does not exceed 100 for Parts 2 and 3.

TIME LIMIT
A time limit of 2 seconds will be placed upon your programme for each test case.

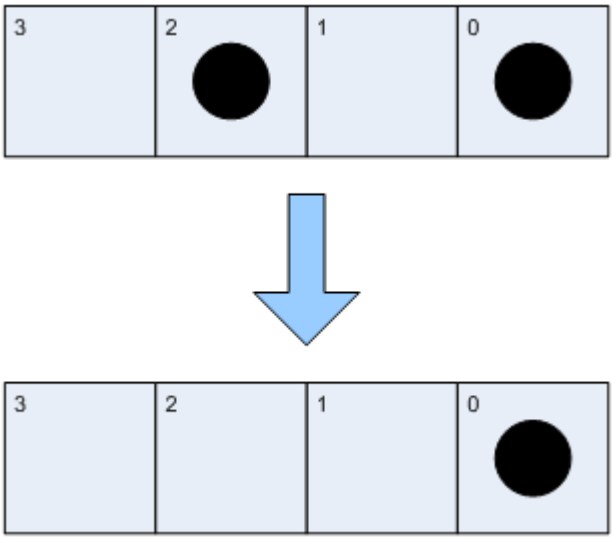
INPUT
Your programme must read from standard input, the following set of data:
- Line 1 contains a single integer indicating the sub-problem: 1 for part 1, 2 for part 2, and 3 for part 3.
- Line 2 contains an integer N, the number of squares on the board.
- Line 3 contains N integers separated by spaces, where the i^{th} number is a value not exceeding 100, indicating the number of pebbles in square *i*.

OUTPUT
Your programme must write to standard output. If Alice has a winning strategy, output 2 lines. In the first line, write "Alice Wins!" without the quotes. In the second line, write three space separated integers *i*, *j*, and *k* representing the move that Alice must make in order to win. If more than one such move exists, you may output any of them.

If Alice does not have a winning strategy, output a single line "Bob Wins!" without the quotes.

EXAMPLE 1

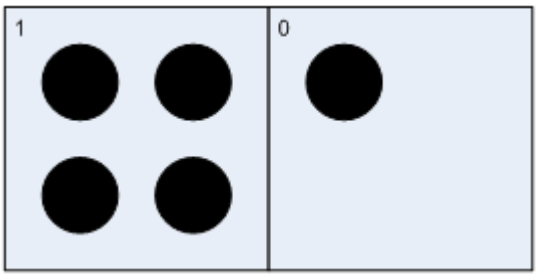
Sample input	Sample output
1 4 1 0 1 0	Alice Wins! 2 1 1



This shows Alice's winning move for the previous game. There is no legal move left for Bob to play.

EXAMPLE 2

Sample input	Sample output
2 2 1 4	Bob Wins!



If played out in full, this game would last exactly 4 moves, because i=1,j=0,k=0 is the only possible move on a 2-square board. After making this move four times (Alice, Bob, Alice, Bob), square 1 will be empty on Alice's move and she will lose.