A. Vasya and Chess **1/2 Point**

2 seconds, 256 megabytes

Vasya decided to learn to play chess. Classic chess doesn't seem interesting to him, so he plays his own sort of chess.

The queen is the piece that captures all squares on its vertical, horizontal and diagonal lines. If the cell is located on the same vertical, horizontal or diagonal line with queen, and the cell contains a piece of the enemy color, the queen is able to move to this square. After that the enemy's piece is removed from the board. The queen cannot move to a cell containing an enemy piece if there is some other piece between it and the queen.

There is an $n \times n$ chessboard. We'll denote a cell on the intersection of the r-th row and c-th column as (r, c). The square $(1, 1)$ contains the white queen and the square $(1, n)$ contains the black queen. All other squares contain green pawns that don't belong to anyone.

The players move in turns. The player that moves first plays for the white queen, his opponent plays for the black queen.

On each move the player has to capture some piece with his queen (that is, move to a square that contains either a green pawn or the enemy queen). The player loses if either he cannot capture any piece during his move or the opponent took his queen during the previous move.

Help Vasya determine who wins if both players play with an optimal strategy on the board $n \times n$.

Input

The input contains a single number n $(2 \le n \le 10^9)$ — the size of the board.

Output

On the first line print the answer to problem — string "`white`" or string "`black`", depending on who wins if the both players play optimally.

If the answer is "`white`", then you should also print two integers r and c representing the cell (r, c), where the first player should make his first move to win. If there are multiple such cells, print the one with the minimum r. If there are still multiple squares, print the one with the minimum c.

input
2
output
white 1 2

input
3
output
black

In the first sample test the white queen can capture the black queen at the first move, so the white player wins.

In the second test from the statement if the white queen captures the green pawn located on the central vertical line, then it will be captured by the black queen during the next move. So the only move for the white player is to capture the green pawn located at $(2, 1)$.

Similarly, the black queen doesn't have any other options but to capture the green pawn located at $(2, 3)$, otherwise if it goes to the middle vertical line, it will be captured by the white queen.

During the next move the same thing happens — neither the white, nor the black queen has other options rather than to capture green pawns situated above them. Thus, the white queen ends up on square $(3, 1)$, and the black queen ends up on square $(3, 3)$.

In this situation the white queen has to capture any of the green pawns located on the middle vertical line, after that it will be captured by the black queen. Thus, the player who plays for the black queen wins.

B. The Game of {a, b, c} Forbidden Take-Away **1 Point**

1 second, 256 megabytes

The game starts with a pile of n stones. Players alternate removing one or more stones from the pile. Any number can be removed except a, b, or c. The winner is the player who makes the last move.

The game, like *nim*, is not that interesting with just one pile. To be able to figure out how to play with multiple piles, you need to compute the *nimber* of a pile. The *nimber* of a game is the *mex* of the *nimbers* of the positions you can reach in one move. Mex is a function that takes a set of non-negative integers and returns the minimum non-negative integer *not* in the set. So, for example, *mex* of the set {0, 1, 2, 4, 8} is 3, and the *nimber* of {5, 7, 8} is 0. (By the way, the *nimber* of a bunch of piles is the xor of the *nimbers* of the individual piles. If it's your turn your goal is to make a move to a position with a *nimber* of 0. But this is beyond the scope of this problem.)

To give a concrete example, let's compute the *nimbers* of different pile sizes in {2, 3} forbidden take-away.

(pile size, *nimber*)

(0, 0): no moves

(2, 0): moves have nimbers {1}

(3, 1): moves have nimbers {0}

(4, 2): moves have nimbers {1, 0}

(5, 3): moves have nimbers {2, 1, 0}

(6, 2): moves have nimbers {3, 0, 1, 0}

In this problem, you will write a program that takes as input a,b,c, and some pile sizes, and will compute the *nimbers* of these pile sizes in {a, b, c} forbidden take-away.

Input

The input consists of up to ten problem instances. Each one begins with a line containing four space-separated positive integers: a,b,c, and k. This is followed by a line containing k space separated pile sizes (the first of which is the largest), for which the *nimber* is to be computed. a, b, and c are all at most 200,000, as are the pile sizes. $k \leq 1000$. The input is terminated by a line with four zeros.

Output

For each input instance, your program should produce a line of output containing k space-separated numbers. These are the *nimbers* of the k pile sizes listed in the input, when playing {a, b, c} forbidden take-away

input
2 3 3 1 6 5 7 9 3 100 76 10 45 78 102 2 500 400 0 0 0 0
output
2 50 36 5 230 220

C. Furlo and Rublo and Game **2 Points**

2 seconds, 256 megabytes

Furlo and Rublo play a game. The table has n piles of coins lying on it, the i-th pile has a_i coins. Furlo and Rublo move in turns, Furlo moves first. In one move you are allowed to:

- choose some pile, let's denote the current number of coins in it as x;
- choose some integer y $(0 \le y < x;\ x^{1/4} \le y \le x^{1/2})$ and decrease the number of coins in this pile to y. In other words, after the described move the pile will have y coins left.

The player who can't make a move, loses.

Your task is to find out, who wins in the given game if both Furlo and Rublo play optimally well.

Input

The first line contains integer n $(1 \le n \le 77777)$ — the number of piles. The next line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 777777777777)$ — the sizes of piles. The numbers are separated by single spaces.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

If both players play optimally well and Furlo wins, print "`Furlo`", otherwise print "`Rublo`". Print the answers without the quotes.

input
1 1
output
Rublo

input
2 1 2
output
Rublo

input
10 1 2 3 4 5 6 7 8 9 10
output
Furlo

D. Game with Powers **2 Points ***

1 second, 256 megabytes

Vasya and Petya wrote down all integers from 1 to n to play the "powers" game (n can be quite large; however, Vasya and Petya are not confused by this fact).

Players choose numbers in turn (Vasya chooses first). If some number x is chosen at the current turn, it is forbidden to choose x or all of its other positive integer powers (that is, x^2, x^3, ...) at the next turns. For instance, if the number 9 is chosen at the first turn, one cannot choose 9 or 81 later, while it is still allowed to choose 3 or 27. The one who cannot make a move loses.

Who wins if both Vasya and Petya play optimally?

Input
Input contains single integer n ($1 \le n \le 10^9$).

Output
Print the name of the winner — "`Vasya`" or "`Petya`" (without quotes).

input
1
output
Vasya

input
2
output
Petya

input
8
output
Petya

In the first sample Vasya will choose 1 and win immediately.

In the second sample no matter which number Vasya chooses during his first turn, Petya can choose the remaining number and win.

E. A Game With Numbers **2.5 Points**

4 seconds, 512 megabytes

Imagine that Alice is playing a card game with her friend Bob. They both have exactly 8 cards and there is an integer on each card, ranging from 0 to 4. In each round, Alice or Bob in turns choose two cards from different players, let them be a and b, where a is the number on the player's card, and b is the number on the opponent's card. It is necessary that $a \cdot b \neq 0$. Then they calculate $c = (a + b) \bmod 5$ and replace the number a with c. The player who ends up with numbers on all 8 cards being 0, wins.

Now Alice wants to know who wins in some situations. She will give you her cards' numbers, Bob's cards' numbers and the person playing the first round. Your task is to determine who wins if both of them choose the best operation in their rounds.

Input

The first line contains one positive integer T ($1 \leq T \leq 100\,000$), denoting the number of situations you need to consider.

The following lines describe those T situations. For each situation:

- The first line contains a non-negative integer f ($0 \leq f \leq 1$), where $f = 0$ means that Alice plays first and $f = 1$ means Bob plays first.
- The second line contains 8 non-negative integers $a_1, a_2, \ldots, a_8$ ($0 \leq a_i \leq 4$), describing Alice's cards.
- The third line contains 8 non-negative integers $b_1, b_2, \ldots, b_8$ ($0 \leq b_i \leq 4$), describing Bob's cards.

We guarantee that if $f = 0$, we have $\sum_{i=1}^{8} a_i \neq 0$. Also when $f = 1$, $\sum_{i=1}^{8} b_i \neq 0$ holds.

Output

Output T lines. For each situation, determine who wins. Output

- "`Alice`" (without quotes) if Alice wins.
- "`Bob`" (without quotes) if Bob wins.
- "`Deal`" (without quotes) if it gets into a deal, i.e. no one wins.

input
4 1 0 0 0 0 0 0 0 0 1 2 3 4 1 2 3 4 1 0 0 0 1 0 0 0 0 0 0 0 0 4 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 4 0 0 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
output
Alice Bob Alice Deal

In the first situation, Alice has all her numbers 0. So she wins immediately.

In the second situation, Bob picks the numbers 4 and 1. Because we have $(4 + 1) \bmod 5 = 0$, Bob wins after this operation.

In the third situation, Alice picks the numbers 1 and 4. She wins after this operation.

In the fourth situation, we can prove that it falls into a loop.

F. An easy problem about trees **3 Points** *

2 seconds, 256 megabytes

Pieguy and Piegirl are playing a game. They have a rooted binary tree, that has a property that each node is either a leaf or has exactly two children. Each leaf has a number associated with it.

On his/her turn a player can choose any two leafs that share their immediate parent, remove them, and associate either of their values with their parent, that now became a leaf (the player decides which of the two values to associate). The game ends when only one node (the one that was the root of the tree) is left.

Pieguy goes first, and his goal is to maximize the value that will be associated with the root when the game ends. Piegirl wants to minimize that value. Assuming that both players are playing optimally, what number will be associated with the root when the game ends?

Input

First line contains a single integer t ($1 \le t \le 100$) — number of test cases. Then t test cases follow. Each test case begins with an empty line, followed by a line with a single integer n ($1 \le n \le 250$), followed by n lines describing n nodes of the tree. Each of those n lines either contains a non-negative number a_i, indicating a leaf node with value a_i ($0 \le a_i \le 1000$) associated with it, or -1 followed by integers l and r, indicating a non-leaf node with children l and r ($0 \le l, r \le n - 1$). Nodes are numbered from 0 to $n - 1$. The root is always node 0.

Output

For each test case print one line with one integer on it — the number that will be associated with the root when the game ends.

input

```
4

3
-1 1 2
10
5

5
-1 1 2
-1 3 4
10
5
20

7
-1 1 2
-1 3 4
-1 5 6
1
2
3
4

11
-1 1 2
-1 3 4
-1 5 6
-1 7 8
15
7
-1 9 10
7
8
9
11
```

output

```
10
10
4
8
```