Bingo

Write a function that simulates a game of Bingo. Specifically, it will take in several Bingo cards, several called squares, and then determine which cards, if any, are winners.

Although the code examples here are given in Java, you may use any programming language you would like.

Note: The rules that are defined here may deviate from the traditional rules of Bingo. Follow the rules as they are laid out here, not as they are defined for Bingo elsewhere in the world.

Here is an example Java function signature for this problem.

```
int[] play(int[][] calledSquares, int[][][] cardData)
```

Input

The function should take two parameters: "called squares" and "card data".

Called Squares

Called squares is a two-dimensional array of integers that represents the squares which will be called during the course of the game of Bingo.

The first dimension of the array represents each round of the game. For example, index 0 of the array is the first round of the game.

The second dimension of the array represents the contents of each called square. It contains exactly two elements with index 0 being the "column" of the square being called and index 1 being the value that would qualify as a match if present in that column in one of the cards.

Card Data

Card data is a three-dimensional array of integers which represents the Bingo cards in play during the game.

The first dimension of the array represents each individual card.

The second dimension of the array represents the row of the given card.

The third dimension of the array represents the value in a particular column of the given row.

Output

The function should return an array of integers representing which cards, if any won the game of Bingo.

Rules

- A winning card is defined as a card where at least one whole row, one whole column, or either diagonal contains values which have been called, as defined in "called squares".
- Cards should be evaluated every turn to see if there are winners. If any winners are identified, the game should end immediately and no further turns should happen.
- It is possible to have more than one winning card if multiple cards are evaluated as winners on the same turn.
- Cards are always square. That is, the number of rows always equals the number of columns.
- It's possible to define "free spaces" in a card by using the number `-1`. Free spaces always evaluate as a match.
- Except for a free space, squares will only contain positive numbers.

Examples

Vertical Match

Input

```
 int[][] calledSquares = new int[][]{
        new int[]{1, 21},
        new int[]{1, 19},
        new int[]{1, 24},
        new int[]{1, 20},
        new int[]{1, 30}
};

int[][][] cardData = new int[][][]{
        new int[][]{
                new int[]{6, 21, 36, 55, 61},
                new int[]{12, 19, 43, 56, 69},
                new int[]{9, 24, -1, 46, 71},
                new int[]{3, 20, 44, 52, 67},
                new int[]{1, 30, 34, 57, 65}
        },
        new int[][]{
                new int[]{4, 16, 40, 46, 72},
                new int[]{10, 17, 41, 58, 62},
                new int[]{2, 26, -1, 48, 66},
                new int[]{7, 18, 37, 60, 63},
                new int[]{14, 30, 35, 59, 73}
        }
};
```

Output

An array indicating that only the first card is a winner. That array will contain the number `0` since the first card is at index `0` of the `cardData` input array.

Free Space Match

Input

```
 int[][] calledSquares = new int[][]{
        new int[]{0, 40},
        new int[]{1, 41},
        new int[]{3, 37},
        new int[]{4, 35}
};

int[][][] cardData = new int[][][]{
        new int[][]{
                new int[]{6, 21, 36, 55, 61},
                new int[]{12, 19, 43, 56, 69},
                new int[]{9, 24, -1, 46, 71},
                new int[]{3, 20, 44, 52, 67},
                new int[]{1, 30, 34, 57, 65}
        },
        new int[][]{
                new int[]{4, 16, 40, 46, 72},
                new int[]{10, 17, 41, 58, 62},
                new int[]{2, 26, -1, 48, 66},
                new int[]{7, 18, 37, 60, 63},
                new int[]{14, 30, 35, 59, 73}
        }
};
```

Output

An array indicating that only the second card is a winner. That array will contain the number `1` since the second card is at index `1` of the `cardData` input array.