```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

    /*
     * Complete the 'spot_the_y' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. INTEGER n
     *  2. STRING line
     */
    public static boolean check(int r, int c, int[][] board, int n, int p) {
        if(r >= n || c >= n || r < 0 || c < 0 || board[r][c] != p) {
            return false;
        }
        return true;
    }
    public static int spot_the_y(int n, String line) {
```

```
    int size = n;

    String[] data = line.split(" ");

    int[] moves = new int[data.length];

    for (int i=0; i<data.length; i++) {

        moves[i] = Integer.parseInt(data[i])-1;

    }

    //System.out.println("moves = "+Arrays.toString(moves));

    int[][] board = new int[size][size];

    int[][] adjacent = {{1,0},{-1,0},{0,1},{0,-1}};

    int[][] diagonal = {{1,1},{1,-1},{-1,1},{-1,-1}};

    boolean p1 = true;

    for (int i=0; i<moves.length; i++) {

        if (p1) {

            int r = moves[i]/size;

            int c = moves[i]%size;

            //System.out.println("moves["+i+"] = "+moves[i]);

            //System.out.println("r = "+r);

            //System.out.println("c = "+c);

            if (board[r][c] == 1) {

                board[r][c] = 0;

            }

            else {

                board[r][c] = 1;

            }

            for (int j=0; j<adjacent.length; j++) {

                if (check(r+adjacent[j][0], c+adjacent[j][1], board, size, 1)) {

                    if (check(r+2*adjacent[j][0]+adjacent[j][1], c+2*adjacent[j][1]+adjacent[j][0], board, size, 1)
&& check(r+2*adjacent[j][0]-adjacent[j][1], c+2*adjacent[j][1]-adjacent[j][0], board, size, 1)) {
```

```
                    return
size*r+c+1+size*(r+2*adjacent[j][0]+adjacent[j][1])+c+2*adjacent[j][1]+adjacent[j][0]+1+size*(r+2*adjac
ent[j][0]-adjacent[j][1])+c+2*adjacent[j][1]-adjacent[j][0]+1+size*(r+adjacent[j][0])+c+adjacent[j][1]+1;

                  }

                }

                if (check(r+diagonal[j][0], c+diagonal[j][1], board, size, 1)) {

                  if (check(r-diagonal[j][0], c, board, size, 1) && check(r+diagonal[j][0], c-diagonal[j][1],
board, size, 1)) {

                    return size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*(r-
diagonal[j][0])+c+1+size*(r+diagonal[j][0])+c-diagonal[j][1]+1;

                  }

                  if (check(r, c-diagonal[j][1], board, size, 1) && check(r-diagonal[j][0], c+diagonal[j][1],
board, size, 1)) {

                    return size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*r+c-
diagonal[j][1]+1+size*(r-diagonal[j][0])+c+diagonal[j][1]+1;

                  }

                  if (check(r+2*diagonal[j][0], c, board, size, 1) && check(r+diagonal[j][0], c+2*diagonal[j][1],
board, size, 1)) {

                    return
size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*(r+2*diagonal[j][0])+c+1+size*(r+diagonal[j][
0])+c+2*diagonal[j][1]+1;

                  }

                  if (check(r, c+2*diagonal[j][1], board, size, 1) && check(r+2*diagonal[j][0], c+diagonal[j][1],
board, size, 1)) {

                    return
size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*r+c+2*diagonal[j][1]+1+size*(r+2*diagonal[j]
[0])+c+diagonal[j][1]+1;

                  }

                }

              }

            }

            else {

              int r = moves[i]/size;
              int c = moves[i]%size;
              //System.out.println("moves["+i+"] = "+moves[i]);
              //System.out.println("r = "+r);
              //System.out.println("c = "+c);
              if (board[r][c] == 2) {
                board[r][c] = 0;
              }
              else {
                board[r][c] = 2;
              }
              for (int j=0; j<adjacent.length; j++) {
                if (check(r+adjacent[j][0], c+adjacent[j][1], board, size, 2)) {
                  if (check(r+2*adjacent[j][0]+adjacent[j][1], c+2*adjacent[j][1]+adjacent[j][0], board, size, 2)
&& check(r+2*adjacent[j][0]-adjacent[j][1], c+2*adjacent[j][1]-adjacent[j][0], board, size, 2)) {
                    return
size*r+c+1+size*(r+2*adjacent[j][0]+adjacent[j][1])+c+2*adjacent[j][1]+adjacent[j][0]+1+size*(r+2*adjac
ent[j][0]-adjacent[j][1])+c+2*adjacent[j][1]-adjacent[j][0]+1+size*(r+adjacent[j][0])+c+adjacent[j][1]+1;
                  }
                }
                if (check(r+diagonal[j][0], c+diagonal[j][1], board, size, 2)) {
                  if (check(r-diagonal[j][0], c, board, size, 2) && check(r+diagonal[j][0], c-diagonal[j][1],
board, size, 2)) {
                    return size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*(r-
diagonal[j][0])+c+1+size*(r+diagonal[j][0])+c-diagonal[j][1]+1;
                  }
                  if (check(r, c-diagonal[j][1], board, size, 2) && check(r-diagonal[j][0], c+diagonal[j][1],
board, size, 2)) {
                    return size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*r+c-
diagonal[j][1]+1+size*(r-diagonal[j][0])+c+diagonal[j][1]+1;
                  }
                  if (check(r+2*diagonal[j][0], c, board, size, 2) && check(r+diagonal[j][0], c+2*diagonal[j][1],
board, size, 2)) {
```

```
                        return
size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*(r+2*diagonal[j][0])+c+1+size*(r+diagonal[j][
0])+c+2*diagonal[j][1]+1;
                    }
                    if (check(r, c+2*diagonal[j][1], board, size, 2) && check(r+2*diagonal[j][0], c+diagonal[j][1],
board, size, 2)) {
                        return
size*r+c+1+size*(r+diagonal[j][0])+c+diagonal[j][1]+1+size*r+c+2*diagonal[j][1]+1+size*(r+2*diagonal[j]
[0])+c+diagonal[j][1]+1;
                    }
                }
            }
        }
        p1 = !p1;
    }
    for (int i=0; i<size; i++) {
        String toPrint = "";
        for (int j=0; j<size; j++) {
            toPrint += board[i][j]+" ";
        }
        System.out.println(toPrint);
    }
    return 0;
  }


}


public class Solution {
  public static void main(String[] args) throws IOException {
    BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
```

```
        BufferedWriter bufferedWriter = new BufferedWriter(new 
FileWriter(System.getenv("OUTPUT_PATH")));


        int n = Integer.parseInt(bufferedReader.readLine().trim());


        String line = bufferedReader.readLine();


        int result = Result.spot_the_y(n, line);

        //System.out.println(result);

        bufferedWriter.write(String.valueOf(result));

        bufferedWriter.newLine();


        bufferedReader.close();

        bufferedWriter.close();

    }

}
```