# Problem C
# Parking Bike

Input file: *testdata.in*
Time limit: 1 seconds

## Problem Description

It is hard to park a bike in campus. Just like rational numbers, the placement of bike also have density: you can always park a bike between any of the two bikes! To prevent from infinite bikes in a parking lot we give some constraints for placement.

We treat every bike as a particle, each parking lot as an interval $(0, 1)$. You can only park a new bike $b_n$ on the valid place of the interval. We define the valid placement as follows:

- 0, 1 are valid place with 0-th level.
- Between any two adjacent valid places $b_l$, $b_r$ with no more than $i-1$-th level, the place $b$ with the ratio of distance from $b$ to $b_l$ and distance from $b$ to $b_r$ is $r : s$, is also a valid placement with $i$-th level. Where $L(b) = \max(L(b_l) + L(b_r)) + 1$, for all $b_l$, $b_r$ such that $L(b_l) < i$ and $L(b_r) < i$.

We now need to check if the placement of bike is valid. For a parking area, given a maximum level $k$, you should check each bike if its level is no more than $k$. If the placement is invalid then you need to move the bike to the nearest valid place.

## Technical Specification

- $1 \leq N \leq 10^6$
- $0 \leq k \leq 10$

- $1 \leq r, s \leq 8$
- $1 \leq m, n < 2^{32}$

## Input Format

There are multiple test cases in the input. The number of test cases will be no more than 100.

The first line of test case contains four integers $N$, $k$, $r$, $s$ respectively. Then followed by $N$ lines, and the $i$-th of the $N$ lines contains the place of the $i$-th bike. The place of bike is represented as a irreducible fraction, as two integers separated by a slash.

## Output Format

For each bike you should output the origin placement. Otherwise you should output the nearest valid placement. If there are two nearest placements then you should output the right one, not the left. The format of placement is same as input: two integers in a line, separated by a slash. Note that 0 and 1 are special cases, their denominator should be 1. You should output 0/1 and 1/1 instead of 0 and 1. Please put a blank line between test cases.

## Sample Input

```
2 0 3 7
3/10
3/5
4 2 1 1
1/2
1/4
1/8
1/16
```

## Sample Output

```
0/1
1/1

1/2
1/4
1/4
0/1
```