Probability has always been an integrated part of computer algorithms. Where the deterministic algorithms have failed to solve a problem in short time, probabilistic algorithms have come to the rescue. In this problem we are not dealing with any probabilistic algorithm. We will just try to determine the winning probability of a certain player.

A game is played by throwing a dice like thing (it should not be assumed that it has six sides like an ordinary dice). If a certain event occurs when a player throws the dice (such as getting a 3, getting green side on top or whatever) he is declared the winner. There can be N such player. So the first player will throw the dice, then the second and at last the N-th player and again the first player and so on. When a player gets the desired event he or she is declared winner and playing stops. You will have to determine the winning probability of one (The I-th) of these players.

Input

Input will contain an integer S ($S \leq 1000$) at first, which indicates how many sets of inputs are there. The next S lines will contain S sets of inputs. Each line contain an integer N ($N \leq 1000$) which denotes the number players, a floating point number p which indicates the probability of the happening of a successful event in a single throw (If success means getting 3 then p is the probability of getting 3 in a single throw. For a normal dice the probability of getting 3 is 1/6), and I ($I \leq N$) the serial of the player whose winning probability is to be determined (Serial no varies from 1 to N). You can assume that no invalid probability (p) value will be given as input.

Output

For each set of input, output in a single line the probability of the I-th player to win. The output floating point number will always have four digits after the decimal point as shown in the sample output.

Sample Input

```
2
2 0.166666 1
2 0.166666 2
```

Sample Output

```
0.5455
0.4545
```