11912 Chutes and Ladders

As the title suggests, this problem is about the popular board game called *Chutes and Ladders*. (For copyright issues it might be inappropriate for us to give a photo of the game board here. Those who really need one may consider using Google.)

In short, the game is played on a 10-by-10 board with squares numbered from 1 to 100. Starting from an "imaginary square 0" outside the board, players take turns using a spinner (or equivalently, a six-faced die) to determine how many squares to move their respective tokens forward. The player who first reaches the last square (i.e. 100) wins. If a move takes the player beyond the square 100, this move is ignored and the token is not moved.

On the game board there are 9 ladders and 10 chutes (or slides). If a token lands on the bottom of a ladder after a move, it is immediately moved to the square at the top of the ladder. Similarly if it lands on the top of a chute, it is moved to the bottom. Here is a list of the ladders and chutes:

- **Ladders**: (1, 38) (4, 14) (9, 31) (21, 42) (28, 84) (36, 44) (51, 67) (71, 91) (80, 100)
- **Chutes**: (16, 6) (47, 26) (49, 11) (56, 53) (62, 19) (64, 60) (87, 24) (93, 73) (95, 75) (98, 78)

The only source of uncertainty of Chutes and Ladders is the results of die rolls (or spinner turns). For an ordinary six-faced fair die, the shortest possible game takes only seven moves, and the average length of a game is around 39 moves. If we use a fair coin as a two-faced die instead (let Head = 1, Tail = 2, for instance), then the corresponding lengths are 15 and 61 respectively. For some "bad" dice like a one-sided die (?!), square 100 may simply be unreachable. We say that a case is *degenerate* if there exists some square reachable from square 0 using the given die, but one can never reach the last square from there.

Write a program to analyse the game if different dice are used.

Input

Each input file contains multiple test cases, and each test case starts on a new line.

A line in the input file starts with an integer n ($1 \leq n \leq 20$), the number of faces on the die. This is followed by n numbers p_1, ..., p_n, with p_k being the probability that a die roll gives a k. Each of these numbers has exactly three decimal places, and of course the probabilities always add up to exactly 1.

Output

For each test case, output the length of the shortest possible game, followed by the average game length (rounded to the nearest integer). For degenerate cases, print the line '`Bad die!`' instead.

Sample Input

```
6 0.167 0.167 0.166 0.166 0.167 0.167
2 0.500 0.500
1 1.000
```

Sample Output

```
7 39
15 61
Bad die!
```