D. Puzzles

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Barney lives in country USC (United States of Charzeh). USC has n cities numbered from 1 through n and n - 1 roads between them. Cities and roads of USC form a rooted tree (Barney's not sure why it is rooted). Root of the tree is the city number 1. Thus if one will start his journey from city 1, he can visit any city he wants by following roads.

Some girl has stolen Barney's heart, and Barney wants to find her. He starts looking for in the root of the tree and (since he is Barney Stinson not a random guy), he uses a *random DFS* to search in the cities. A
pseudo code of this algorithm is as follows:

```
let starting_time be an array of length n
current_time = 0
dfs(v):
        current_time = current_time + 1
        starting_time[v] = current_time
        shuffle children[v] randomly (each permutation with equal possibility)
        // children[v] is vector of children cities of city v
        for u in children[v]:
                dfs(u)
```

As told before, Barney will start his journey in the root of the tree (equivalent to call `dfs(1)`).

Now Barney needs to pack a backpack and so he wants to know more about his upcoming journey: for every city i, Barney wants to know the expected value of `starting_time[i]`. He's a friend of Jon Snow and knows nothing, that's why he asked for your help.

Input

The first line of input contains a single integer n ($1 \le n \le 10^5$) — the number of cities in USC.

The second line contains n - 1 integers $p_2, p_3, ..., p_n$ ($1 \le p_i < i$), where p_i is the number of the parent city of city number i in the tree, meaning there is a road between cities numbered p_i and i in USC.

Output

In the first and only line of output print n numbers, where i-th number is the expected value of `starting_time[i]`.

Your answer for each city will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Examples

input
7 1 2 1 1 4 4

output
1.0 4.0 5.0 3.5 4.5 5.0 5.0

input
12 1 1 2 2 4 4 3 3 1 10 8

output

```
1.0 5.0 5.5 6.5 7.5 8.0 8.0 7.0 7.5 6.5 7.5 8.0
```