E. Eyes Closed

time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vasya and Petya were tired of studying so they decided to play a game. Before the game begins Vasya looks at array a consisting of n integers. As soon as he remembers all elements of a the game begins. Vasya closes his eyes and Petya does q actions of one of two types:

$1)$ Petya says 4 integers $l1, r1, l2, r2$ — boundaries of two non-intersecting segments. After that he swaps one random element from the $[l1, r1]$ segment with another random element from the $[l2, r2]$ segment.

$2)$ Petya asks Vasya the sum of the elements of a in the $[l, r]$ segment.

Vasya is a mathematician so he answers Petya the mathematical expectation of the sum of the elements in the segment.

Your task is to write a program which will answer the second type questions as Vasya would do it. In other words your program should print the mathematical expectation of the sum of the elements of a in the $[l, r]$ segment for every second type query.

Input

The first line contains two integers n, q $(2 \le n \le 10^5, 1 \le q \le 10^5)$ — the number of elements in the array and the number of queries you need to handle.

The second line contains n integers a_i $(1 \le a_i \le 10^9)$ — elements of the array.

The next q lines contain Petya's actions of type 1 or 2.

If it is a type 1 action then the line contains 5 integers $1, l1, r1, l2, r2$ $(1 \le l1 \le r1 \le n, 1 \le l2 \le r2 \le n)$.

If it is a type 2 query then the line contains 3 integers $2, l, r$ $(1 \le l \le r \le n)$.

It is guaranteed that there is at least one type 2 query and segments $[l1, r1], [l2, r2]$ don't have common elements.

Output

For each type 2 query print one real number — the mathematical expectation of the sum of elements in the segment.

Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-4} — formally, the answer is correct if where x is jury's answer and y is yours.

Examples

input
4 4 1 1 2 2 1 2 2 3 3 2 1 2 1 1 2 3 4 2 1 2

output
3.0000000 3.0000000

input

```
10 5
1 1 1 1 1 2 2 2 2 2
1 1 5 6 10
2 1 5
1 1 5 6 10
1 1 5 6 10
2 6 10
```

output

```
6.0000000
8.0400000
```

input

```
10 10
1 2 3 4 5 6 7 8 9 10
1 1 5 6 10
1 1 5 6 10
2 1 5
1 1 3 6 9
2 1 3
1 5 7 8 10
1 1 1 10 10
2 1 5
2 7 10
2 1 10
```

output

```
23.0000000
14.0000000
28.0133333
21.5733333
55.0000000
```