# A. Sereja and Prefixes

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Sereja loves number sequences very much. That's why he decided to make himself a new one following a certain algorithm.

Sereja takes a blank piece of paper. Then he starts writing out the sequence in $m$ stages. Each time he either adds a new number to the end of the sequence or takes $l$ first elements of the current sequence and adds them $c$ times to the end. More formally, if we represent the current sequence as $a_1, a_2, ..., a_n$, then after we apply the described operation, the sequence transforms into $a_1, a_2, ..., a_n[, a_1, a_2, ..., a_l]$ (the block in the square brackets must be repeated $c$ times).

A day has passed and Sereja has completed the sequence. He wonders what are the values of some of its elements. Help Sereja.

## Input

The first line contains integer $m$ $(1 \le m \le 10^5)$ — the number of stages to build a sequence.

Next $m$ lines contain the description of the stages in the order they follow. The first number in the line is a type of stage (1 or 2). Type 1 means adding one number to the end of the sequence, in this case the line contains integer $x_i$ $(1 \le x_i \le 10^5)$ — the number to add. Type 2 means copying a prefix of length $l_i$ to the end $c_i$ times, in this case the line further contains two integers $l_i, c_i$ $(1 \le l_i \le 10^5, 1 \le c_i \le 10^4)$, $l_i$ is the length of the prefix, $c_i$ is the number of copyings. It is guaranteed that the length of prefix $l_i$ is never larger than the current length of the sequence.

The next line contains integer $n$ $(1 \le n \le 10^5)$ — the number of elements Sereja is interested in. The next line contains the numbers of elements of the final sequence Sereja is interested in. The numbers are given in the strictly increasing order. It is guaranteed that all numbers are strictly larger than zero and do not exceed the length of the resulting sequence. Consider the elements of the final sequence numbered starting from $1$ from the beginning to the end of the sequence.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

## Output

Print the elements that Sereja is interested in, in the order in which their numbers occur in the input.

**Sample test(s)**

input

```
6
1 1
1 2
2 2 1
1 3
2 5 2
1 4
16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

output

```
1 2 1 2 3 1 2 1 2 3 1 2 1 2 3 4
```

# B. Sereja and Tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sereja adores trees. Today he came up with a revolutionary new type of binary root trees.

His new tree consists of $n$ levels, each vertex is indexed by two integers: the number of the level and the number of the vertex on the current level. The tree root is at level $1$, its index is $(1, 1)$. Here is a pseudo code of tree construction.

```
//the global data are integer arrays cnt[], left[][], right[][]

cnt[1] = 1;
fill arrays left[][], right[][] with values -1;
for(level = 1; level < n; level = level + 1){
    cnt[level + 1] = 0;
    for(position = 1; position <= cnt[level]; position = position + 1){
        if(the value of position is a power of two){ // that is, 1, 2, 4, 8...
            left[level][position] = cnt[level + 1] + 1;
            right[level][position] = cnt[level + 1] + 2;
            cnt[level + 1] = cnt[level + 1] + 2;
        }else{
            right[level][position] = cnt[level + 1] + 1;
            cnt[level + 1] = cnt[level + 1] + 1;
        }
    }
}
```

After the pseudo code is run, cell `cnt[level]` contains the number of vertices on level $level$. Cell `left[level][position]` contains the number of the vertex on the level $level + 1$, which is the left child of the vertex with index $(level, position)$, or it contains -1, if the vertex doesn't have a left child. Similarly, cell `right[level][position]` is responsible for the right child. You can see how the tree with $n = 4$ looks like in the notes.

Serja loves to make things complicated, so he first made a tree and then added an empty set $A(level, position)$ for each vertex. Then Sereja executes $m$ operations. Each operation is of one of the two following types:

- The format of the operation is "$1\ t\ l\ r\ x$". For all vertices $level, position$ $(level = t;\ l \le position \le r)$ add value $x$ to set $A(level, position)$.
- The format of the operation is "$2\ t\ v$". For vertex $level, position$ $(level = t, position = v)$, find the union of all sets of vertices that are in the subtree of vertex $(level, position)$. Print the size of the union of these sets.

Help Sereja execute the operations. In this problem a set contains only distinct values like std::set in C++.

### Input

The first line contains integers $n$ and $m$ $(1 \le n, m \le 7000)$.

Next $m$ lines contain the descriptions of the operations. The operation of the first type is given by five integers: $1\ t\ l\ r\ x$ $(1 \le t \le n;\ 1 \le l \le r \le cnt[t];\ 1 \le x \le 10^6)$. The operation of the second type is given by three integers: $2\ t\ v$ $(1 \le t \le n;\ 1 \le v \le cnt[t])$.

### Output

For each operation of the second type, print the answer on a single line.

#### Sample test(s)

input

```
4 5
1 4 4 7 1
1 3 1 2 2
2 1 1
2 4 1
2 3 3
```

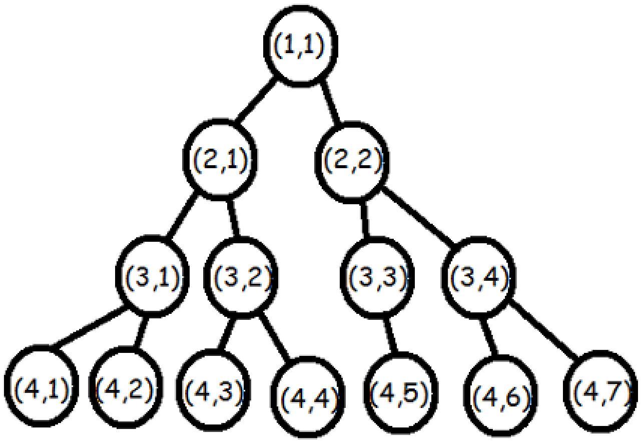
output

```
2
0
1
```

### Note

You can find the definitions that are used while working with root trees by this link: `http://en.wikipedia.org/wiki/Tree_(graph_theory).`

You can see an example of a constructed tree at $n = 4$ below.


(1,1)
(2,1)
(2,2)
(3,1)
(3,2)
(3,3)
(3,4)
(4,1)
(4,2)
(4,3)
(4,4)
(4,5)
(4,6)
(4,7)

# C. Sereja and Brackets

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

Sereja has a bracket sequence $s_1, s_2, ..., s_n$, or, in other words, a string $s$ of length $n$, consisting of characters "`(`" and "`)`".

Sereja needs to answer $m$ queries, each of them is described by two integers $l_i, r_i$ $(1 \le l_i \le r_i \le n)$. The answer to the $i$-th query is the length of the maximum correct bracket subsequence of sequence $s_{l_i}, s_{l_i+1}, ..., s_{r_i}$. Help Sereja answer all queries.

You can find the definitions for a subsequence and a correct bracket sequence in the notes.

## Input

The first line contains a sequence of characters $s_1, s_2, ..., s_n$ $(1 \le n \le 10^6)$ without any spaces. Each character is either a "`(`" or a "`)`". The second line contains integer $m$ $(1 \le m \le 10^5)$ — the number of queries. Each of the next $m$ lines contains a pair of integers. The $i$-th line contains integers $l_i, r_i$ $(1 \le l_i \le r_i \le n)$ — the description of the $i$-th query.

## Output

Print the answer to each question on a single line. Print the answers in the order they go in the input.

**Sample test(s)**

input

```
())(())(())(
7
1 1
2 3
1 2
1 12
8 12
5 11
2 10
```

output

```
0
0
2
10
4
6
6
```

## Note

A *subsequence* of length $|x|$ of string $s = s_1 s_2 ... s_{|s|}$ (where $|s|$ is the length of string $s$) is string $x = s_{k_1} s_{k_2} ... s_{k_{|x|}}$ $(1 \le k_1 < k_2 < ... < k_{|x|} \le |s|)$.

A *correct bracket sequence* is a bracket sequence that can be transformed into a correct aryphmetic expression by inserting characters "`1`" and "`+`" between the characters of the string. For example, bracket sequences "`()()`", "`(())`" are correct (the resulting expressions "`(1)+(1)`", "`((1+1)+1)`"), and "`)(`" and "`(`" are not.

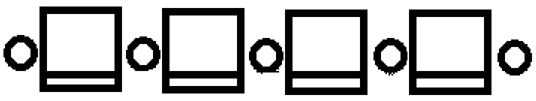
For the third query required sequence will be «`()`».

For the fourth query required sequence will be «`()(())(())`».

# D. Sereja and Cinema

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

The cinema theater hall in Sereja's city is $n$ seats lined up in front of one large screen. There are slots for personal possessions to the left and to the right of each seat. Any two adjacent seats have exactly one shared slot. The figure below shows the arrangement of seats and slots for $n = 4$.



Today it's the premiere of a movie called "Dry Hard". The tickets for all the seats have been sold. There is a very strict controller at the entrance to the theater, so all $n$ people will come into the hall one by one. As soon as a person enters a cinema hall, he immediately (momentarily) takes his seat and occupies all empty slots to the left and to the right from him. If there are no empty slots, the man gets really upset and leaves.

People are not very constant, so it's hard to predict the order in which the viewers will enter the hall. For some seats, Sereja knows the number of the viewer (his number in the entering queue of the viewers) that will come and take this seat. For others, it can be any order.

Being a programmer and a mathematician, Sereja wonders: how many ways are there for the people to enter the hall, such that nobody gets upset? As the number can be quite large, print it modulo $1000000007$ $(10^9 + 7)$.

### Input

The first line contains integer $n$ $(1 \le n \le 10^5)$. The second line contains $n$ integers, the $i$-th integer shows either the index of the person (index in the entering queue) with the ticket for the $i$-th seat or a $0$, if his index is not known. It is guaranteed that all positive numbers in the second line are distinct.

You can assume that the index of the person who enters the cinema hall is a unique integer from 1 to $n$. The person who has index 1 comes first to the hall, the person who has index 2 comes second and so on.

### Output

In a single line print the remainder after dividing the answer by number $1000000007$ $(10^9 + 7)$.

**Sample test(s)**

input

```
11
0 0 0 0 0 0 0 0 0 0 0
```

output

```
1024
```

input

```
6
0 3 1 0 0 0
```

output

```
3
```

# E. Sereja and Dividing

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's assume that we have a sequence of doubles $a_1, a_2, ..., a_{|a|}$ and a double variable $x$. You are allowed to perform the following two-staged operation:

1. choose an index of the sequence element $i$ $(1 \le i \le |a|)$;
2. consecutively perform assignments: $tmp = \frac{a_i + x}{2}, a_i = tmp, x = tmp$.

Let's use function $g(a, x)$ to represent the largest value that can be obtained from variable $x$, using the described operation any number of times and sequence $a$.

Sereja has sequence $b_1, b_2, ..., b_{|b|}$. Help Sereja calculate sum: $\sum_{i=1}^{|b|} \sum_{j=i}^{|b|} g([b_i, b_{i+1}, \ldots, b_j], 0)$. Record $[b_i, b_{i+1}, ..., b_j]$ represents a sequence containing the elements in brackets in the given order. To avoid problems with precision, please, print the required sum divided by $|b|^2$.

## Input

The first line contains integer $|b|$ $(1 \le |b| \le 3 \cdot 10^5)$ — the length of sequence $b$. The second line contains $|b|$ integers $b_1, b_2, ..., b_{|b|}$ $(1 \le b_i \le 10^5)$.

## Output

In a single line print a real number — the required sum divided by $|b|^2$. Your answer will be considered correct if its absolute or relative error won't exceed $10^{-6}$.

**Sample test(s)**

input

```
5
1 2 3 4 1
```

output

```
1.238750000000000
```