Codeforces Round #223 (Div. 2)

A. Sereja and Dima

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

Sereja and Dima play a game. The rules of the game are very simple. The players have n cards in a row. Each card contains a number, all numbers on the cards are distinct. The players take turns, Sereja moves first. During his turn a player can take one card: either the leftmost card in a row, or the rightmost one. The game ends when there is no more cards. The player who has the maximum sum of numbers on his cards by the end of the game, wins.

Sereja and Dima are being greedy. Each of them chooses the card with the larger number during his move.

Inna is a friend of Sereja and Dima. She knows which strategy the guys are using, so she wants to determine the final score, given the initial state of the game. Help her.

Input

The first line contains integer n $(1 \le n \le 1000)$ — the number of cards on the table. The second line contains space-separated numbers on the cards from left to right. The numbers on the cards are distinct integers from 1 to 1000.

Output

On a single line, print two integers. The first number is the number of Sereja's points at the end of the game, the second number is the number of Dima's points at the end of the game.

Sample test(s)

input
4 4 1 2 10
output
12 5

input
7 1 2 3 4 5 6 7
output
16 12

Note

In the first sample Sereja will take cards with numbers 10 and 2, so Sereja's sum is 12. Dima will take cards with numbers 4 and 1, so Dima's sum is 5.

B. Sereja and Stairs

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

Sereja loves integer sequences very much. He especially likes stairs.

Sequence $a_1, a_2, ..., a_{|a|}$ ($|a|$ is the length of the sequence) is stairs if there is such index i $(1 \le i \le |a|)$, that the following condition is met:

$$a_1 < a_2 < ... < a_{i-1} < a_i > a_{i+1} > ... > a_{|a|-1} > a_{|a|}.$$

For example, sequences [1, 2, 3, 2] and [4, 2] are stairs and sequence [3, 1, 2] isn't.

Sereja has m cards with numbers. He wants to put some cards on the table in a row to get a stair sequence. What maximum number of cards can he put on the table?

Input

The first line contains integer m $(1 \le m \le 10^5)$ — the number of Sereja's cards. The second line contains m integers b_i $(1 \le b_i \le 5000)$ — the numbers on the Sereja's cards.

Output

In the first line print the number of cards you can put on the table. In the second line print the resulting stairs.

Sample test(s)

input
5 1 2 3 4 5
output
5 5 4 3 2 1

input
6 1 1 2 2 3 3
output
5 1 2 3 2 1

C. 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

D. 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)

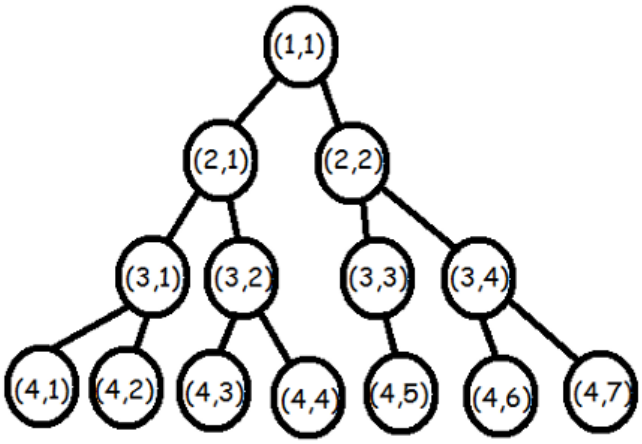
input
4 5 1 4 4 7 1 1 3 1 2 2 2 1 1 2 4 1 2 3 3

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)

E. 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 «()(())(())».