Codeforces Round #276 (Div. 1)

A. Bits

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

Let's denote as $\text{popcount}(x)$ the number of bits set ('1' bits) in the binary representation of the non-negative integer x.

You are given multiple queries consisting of pairs of integers l and r. For each query, find the x, such that $l \le x \le r$, and $\text{popcount}(x)$ is maximum possible. If there are multiple such numbers find the smallest of them.

Input

The first line contains integer n — the number of queries ($1 \le n \le 10000$).

Each of the following n lines contain two integers l_i, r_i — the arguments for the corresponding query ($0 \le l_i \le r_i \le 10^{18}$).

Output

For each query print the answer in a separate line.

Examples

input
3 1 2 2 4 1 10
output
1 3 7

Note

The binary representations of numbers from 1 to 10 are listed below:

$1_{10} = 1_2$

$2_{10} = 10_2$

$3_{10} = 11_2$

$4_{10} = 100_2$

$5_{10} = 101_2$

$6_{10} = 110_2$

$7_{10} = 111_2$

$8_{10} = 1000_2$

$9_{10} = 1001_2$

$10_{10} = 1010_2$

B. Maximum Value

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

You are given a sequence a consisting of n integers. Find the maximum possible value of $a_i \bmod a_j$ (integer remainder of a_i divided by a_j), where $1 \le i, j \le n$ and $a_i \ge a_j$.

Input

The first line contains integer n — the length of the sequence ($1 \le n \le 2 \cdot 10^5$).

The second line contains n space-separated integers a_i ($1 \le a_i \le 10^6$).

Output

Print the answer to the problem.

Examples

input
3 3 4 5
output
2

C. Strange Sorting

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

How many specific orders do you know? Ascending order, descending order, order of ascending length, order of ascending polar angle... Let's have a look at another specific order: d-*sorting*. This sorting is applied to the strings of length at least d, where d is some positive integer. The characters of the string are sorted in following manner: first come all the 0-th characters of the initial string, then the 1-st ones, then the 2-nd ones and so on, in the end go all the $(d - 1)$-th characters of the initial string. By the i-th characters we mean all the character whose positions are exactly i modulo d. If two characters stand on the positions with the same remainder of integer division by d, their relative order after the sorting shouldn't be changed. The string is zero-indexed. For example, for string 'qwerty':

Its 1-sorting is the string 'qwerty' (all characters stand on 0 positions),

Its 2-sorting is the string 'qetwry' (characters 'q', 'e' and 't' stand on 0 positions and characters 'w', 'r' and 'y' are on 1 positions),

Its 3-sorting is the string 'qrwtey' (characters 'q' and 'r' stand on 0 positions, characters 'w' and 't' stand on 1 positions and characters 'e' and 'y' stand on 2 positions),

Its 4-sorting is the string 'qtwyer',

Its 5-sorting is the string 'qywert'.

You are given string S of length n and m *shuffling* operations of this string. Each *shuffling* operation accepts two integer arguments k and d and transforms string S as follows. For each i from 0 to $n - k$ in the increasing order we apply the operation of d-sorting to the substring $S[i..i + k - 1]$. Here $S[a..b]$ represents a substring that consists of characters on positions from a to b inclusive.

After each *shuffling* operation you need to print string S.

Input

The first line of the input contains a non-empty string S of length n, consisting of lowercase and uppercase English letters and digits from 0 to 9.

The second line of the input contains integer m – the number of *shuffling* operations ($1 \le m \cdot n \le 10^6$).

Following m lines contain the descriptions of the operations consisting of two integers k and d ($1 \le d \le k \le n$).

Output

After each operation print the current state of string S.

Examples

input
qwerty 3 4 2 6 3 5 2
output
qertwy qtewry qetyrw

Note

Here is detailed explanation of the sample. The first modification is executed with arguments $k = 4$, $d = 2$. That means that you need to apply 2-sorting for each substring of length 4 one by one moving from the left to the right. The string will transform in the following manner:

qwerty → **qewr**ty → q**erwt**y → qe**rtwy**

Thus, string S equals 'qertwy' at the end of first query.

The second modification is executed with arguments $k = 6$, $d = 3$. As a result of this operation the whole string S is replaced by its 3-sorting:

qertwy → **qtewry**

The third modification is executed with arguments $k = 5$, $d = 2$.

qtewry → **qertwy** → q**etyrw**

D. Kindergarten

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

In a kindergarten, the children are being divided into groups. The teacher put the children in a line and associated each child with his or her integer charisma value. Each child should go to exactly one group. Each group should be a nonempty segment of consecutive children of a line. A group's *sociability* is the maximum difference of charisma of two children in the group (in particular, if the group consists of one child, its sociability equals a zero).

The teacher wants to divide the children into some number of groups in such way that the total *sociability* of the groups is maximum. Help him find this value.

Input

The first line contains integer n — the number of children in the line ($1 \le n \le 10^6$).

The second line contains n integers a_i — the charisma of the i-th child ($-10^9 \le a_i \le 10^9$).

Output

Print the maximum possible total sociability of all groups.

Examples

input
5 1 2 3 1 2
output
3

input
3 3 3 3
output
0

Note

In the first test sample one of the possible variants of an division is following: the first three children form a group with sociability 2, and the two remaining children form a group with sociability 1.

In the second test sample any division leads to the same result, the sociability will be equal to 0 in each group.

E. Sign on Fence

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bizon the Champion has recently finished painting his wood fence. The fence consists of a sequence of n panels of 1 meter width and of arbitrary height. The i-th panel's height is h_i meters. The adjacent planks follow without a gap between them.

After Bizon painted the fence he decided to put a "for sale" sign on it. The sign will be drawn on a rectangular piece of paper and placed on the fence so that the sides of the sign are parallel to the fence panels and are also aligned with the edges of some panels. Bizon the Champion introduced the following constraints for the sign position:

1. The width of the sign should be exactly w meters.
2. The sign must fit into the segment of the fence from the l-th to the r-th panels, inclusive (also, it can't exceed the fence's bound in vertical direction).

The sign will be really pretty, So Bizon the Champion wants the sign's height to be as large as possible.

You are given the description of the fence and several queries for placing sign. For each query print the maximum possible height of the sign that can be placed on the corresponding segment of the fence with the given fixed width of the sign.

Input

The first line of the input contains integer n — the number of panels in the fence ($1 \le n \le 10^5$).

The second line contains n space-separated integers h_i, — the heights of the panels ($1 \le h_i \le 10^9$).

The third line contains an integer m — the number of the queries ($1 \le m \le 10^5$).

The next m lines contain the descriptions of the queries, each query is represented by three integers l, r and w ($1 \le l \le r \le n$, $1 \le w \le r - l + 1$) — the segment of the fence and the width of the sign respectively.

Output

For each query print the answer on a separate line — the maximum height of the sign that can be put in the corresponding segment of the fence with all the conditions being satisfied.

Examples

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

Note

The fence described in the sample looks as follows:

The possible positions for the signs for all queries are given below.

The optimal position of the sign for the first query.

The optimal position of the sign for the second query.

The optimal position of the sign for the third query.