A. Perpendicular Segments

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given a coordinate plane and three integers X, Y, and K. Find two line segments AB and CD such that

1. the coordinates of points A, B, C, and D are integers;
2. $0 \le A_x, B_x, C_x, D_x \le X$ and $0 \le A_y, B_y, C_y, D_y \le Y$;
3. the length of segment AB is at least K;
4. the length of segment CD is at least K;
5. segments AB and CD are *perpendicular*: if you draw lines that contain AB and CD, they will cross at a right angle.

Note that it's **not** necessary for segments to intersect. Segments are perpendicular as long as the lines they induce are perpendicular.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. Next, t cases follow.

The first and only line of each test case contains three integers X, Y, and K ($1 \le X, Y \le 1000$; $1 \le K \le 1414$).

Additional constraint on the input: the values of X, Y, and K are chosen in such a way that the answer exists.

Output

For each test case, print two lines. The first line should contain 4 integers A_x, A_y, B_x, and B_y — the coordinates of the first segment.

The second line should also contain 4 integers C_x, C_y, D_x, and D_y — the coordinates of the second segment.

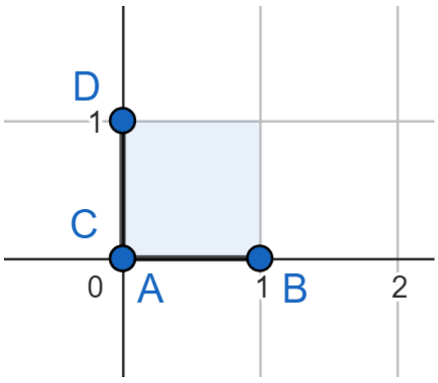
If there are multiple answers, print any of them.

Standard Input	Standard Output

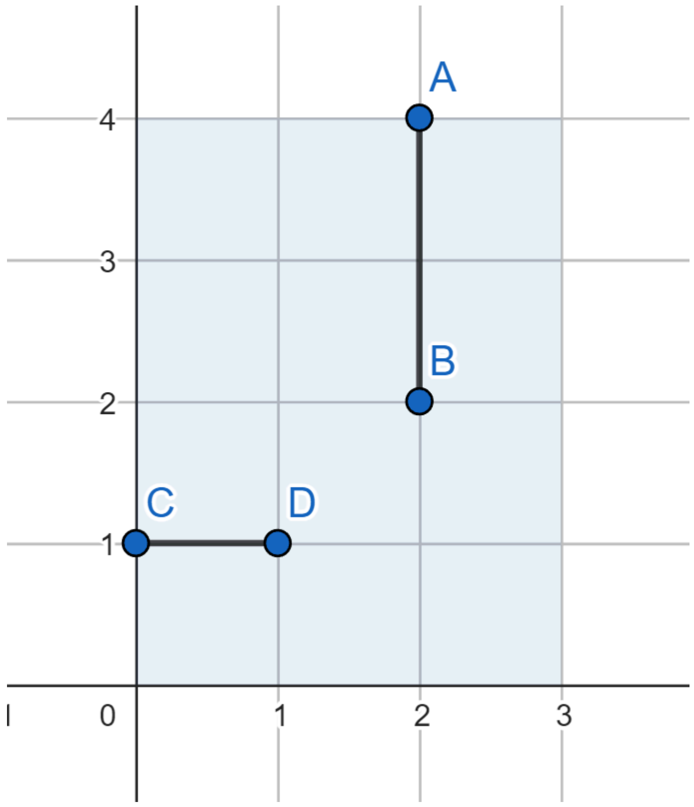
4
1 1 1
3 4 1
4 3 3
3 4 4
	0 0 1 0
	0 0 0 1
	2 4 2 2
	0 1 1 1
	0 0 1 3
	1 2 4 1
	0 1 3 4
	0 3 3 0

Note

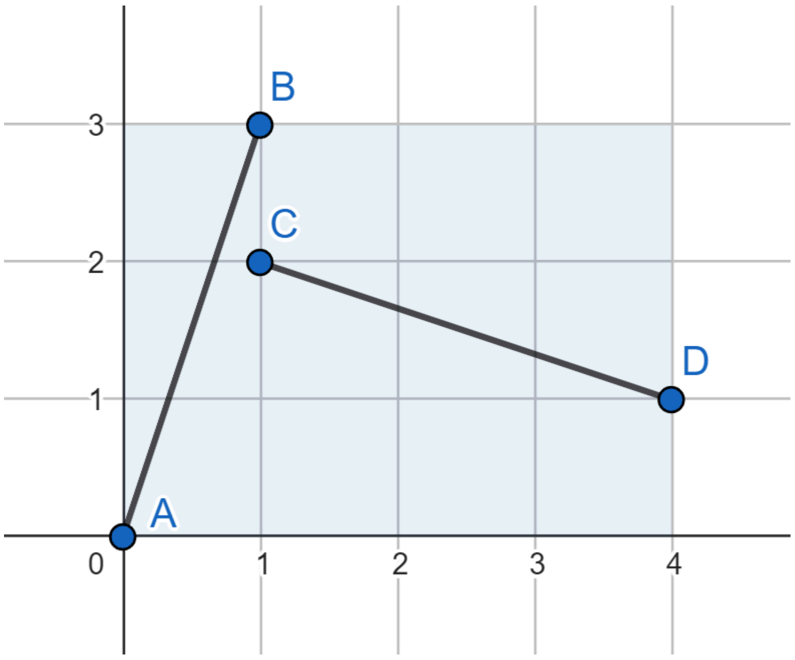
The answer for the first test case is shown below:



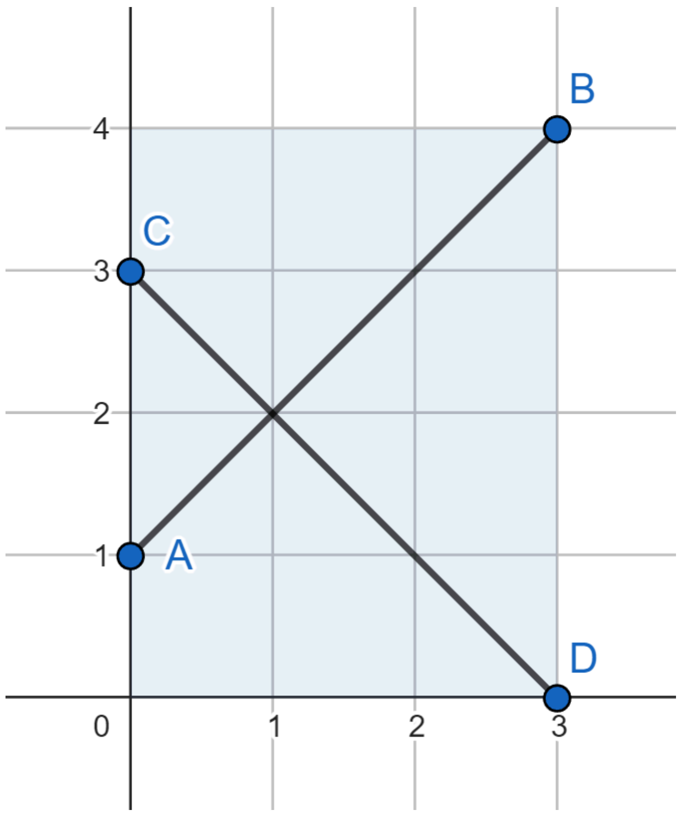
The answer for the second test case:



The answer for the third test case:



The answer for the fourth test case:



B
C
A
D
4
3
2
1
0
1
2
3

,

B. Black Cells

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given a strip divided into cells, numbered from left to right from 0 to 10^{18}. Initially, all cells are white.

You can perform the following operation: choose two **white** cells i and j, such that $i \neq j$ and $|i - j| \leq k$, and paint them black.

A list a is given. All cells from this list must be painted black. Additionally, **at most one** cell that is not in this list can also be painted black. Your task is to determine the minimum value of k for which this is possible.

Input

The first line contains a single integer t ($1 \leq t \leq 500$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 2000$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 < a_i < 10^{18}$; $a_i < a_{i+1}$).

Additional constraint on the input: the sum of n across all test cases does not exceed 2000.

Output

For each test case, print a single integer — the minimum value of k for which it is possible to paint all the given cells black.

Standard Input	Standard Output

4
2
1 2
1
7
3
2 4 9
5
1 5 8 10 13
	1
	1
	2
	3

Note

In the first example, with $k = 1$, it is possible to paint the cells $(1, 2)$.

In the second example, with $k = 1$, it is possible to paint the cells $(7, 8)$.

In the third example, with $k = 2$, it is possible to paint the cells $(2, 4)$ and $(8, 9)$.

In the fourth example, with $k = 3$, it is possible to paint the cells $(0, 1)$, $(5, 8)$ and $(10, 13)$.

,

C. Action Figures

```
Input file:     standard input
Output file:    standard output
Time limit:     2.5 seconds
Memory limit:   512 megabytes
```

There is a shop that sells action figures near Monocarp's house. A new set of action figures will be released shortly; this set contains n figures, the i-th figure costs i coins and is available for purchase from day i to day n.

For each of the n days, Monocarp knows whether he can visit the shop.

Every time Monocarp visits the shop, he can buy any number of action figures which are sold in the shop (of course, he cannot buy an action figure that is not yet available for purchase). If Monocarp buys **at least two** figures during the same day, he gets a discount equal to the cost of **the most expensive** figure he buys (in other words, he gets the most expensive of the figures he buys for free).

Monocarp wants to buy **exactly** one 1-st figure, one 2-nd figure, ..., one n-th figure from the set. He cannot buy the same figure twice. What is the minimum amount of money he has to spend?

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case consists of two lines:

- the first line contains one integer n ($1 \le n \le 4 \cdot 10^5$) — the number of figures in the set (and the number of days);
- the second line contains a string s ($|s| = n$, each s_i is either `0` or `1`). If Monocarp can visit the shop on the i-th day, then s_i is `1`; otherwise, s_i is `0`.

Additional constraints on the input:

- in each test case, s_n is `1`, so Monocarp is always able to buy all figures during the n-th day;
- the sum of n over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case, print one integer — the minimum amount of money Monocarp has to spend.

Standard Input	Standard Output
4 1 1 6 101101 7 1110001 5 11111	1 8 18 6

Note

In the first test case, Monocarp buys the 1-st figure on the 1-st day and spends 1 coin.

In the second test case, Monocarp can buy the 1-st and the 3-rd figure on the 3-rd day, the 2-nd and the 4-th figure on the 4-th day, and the 5-th and the 6-th figure on the 6-th day. Then, he will spend $1 + 2 + 5 = 8$ coins.

In the third test case, Monocarp can buy the 2-nd and the 3-rd figure on the 3-rd day, and all other figures on the 7-th day. Then, he will spend $1 + 2 + 4 + 5 + 6 = 18$ coins.

,

D. Sums of Segments

Input file:	standard input
Output file:	standard output
Time limit:	4 seconds
Memory limit:	512 megabytes

You are given a sequence of integers $[a_1, a_2, \dots, a_n]$. Let $s(l, r)$ be the sum of elements from a_l to a_r (i. e. $s(l, r) = \sum_{i=l}^{r} a_i$).

Let's construct another sequence b of size $\frac{n(n+1)}{2}$ as follows: $b = [s(1,1), s(1,2), \dots, s(1,n), s(2,2), s(2,3), \dots, s(2,n), s(3,3), \dots, s(n,n)]$.

For example, if $a = [1, 2, 5, 10]$, then $b = [1, 3, 8, 18, 2, 7, 17, 5, 15, 10]$.

You are given q queries. During the i-th query, you are given two integers l_i and r_i, and you have to calculate $\sum_{j=l_i}^{r_i} b_j$.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($-10 \le a_i \le 10$).

The third line contains one integer q ($1 \le q \le 3 \cdot 10^5$).

Then q lines follow, the i-th of them contains two integers l_i and r_i ($1 \le l_i \le r_i \le \frac{n(n+1)}{2}$).

Output

Print q integers, the i-th of which should be equal to $\sum_{j=l_i}^{r_i} b_j$.

Standard Input	Standard Output

4
1 2 5 10
15
1 1
1 2
1 3
1 4
1 5
1 10
5 10
6 10
2 8
3 4
3 10
3 8
5 6
	1
	4
	12
	30
	32
	86
	56
	54
	60
	26
	82
	57
	9
	2
	61

5 5 1 8	

E. Best Subsequence

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Given an integer array a of size n.

Let's define the *value* of the array as its size minus the number of set bits in the bitwise OR of all elements of the array.

For example, for the array $[1, 0, 1, 2]$, the bitwise OR is 3 (which contains 2 set bits), and the value of the array is $4 - 2 = 2$.

Your task is to calculate the maximum possible value of some subsequence of the given array.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 100$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{60}$).

Output

For each test case, print the maximum possible value of some subsequence of the given array.

Standard Input	Standard Output
4	3
3	2
0 0 0	0
4	3
1 0 1 2	
1	
5	
8	
7 1 48 14 13 8 7 6	

F. Bermart Ice Cream

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   1024 megabytes
```

In the Bermart chain of stores, a variety of ice cream is sold. Each type of ice cream has two parameters: price and tastiness.

Initially, there is one store numbered 1, which sells nothing. You have to process q queries of the following types:

- $1\ x$ — a new store opens, that sells the same types of ice cream as store x. It receives the minimum available positive index. The order of the types of ice cream in the new store is the same as in store x.
- $2\ x\ p\ t$ — a type of ice cream with price p and tastiness t becomes available in store x.
- $3\ x$ — a type of ice cream that was available the longest (appeared the earliest) in store x is removed.
- $4\ x\ p$ — for store x, find the maximum total tastiness of a subset of types of ice cream that are sold there, such that the total price does not exceed p (each type can be used in the subset no more than once).

Input

The first line contains a single integer q ($1 \le q \le 3 \cdot 10^4$) — the number of queries.

Each of the following q lines contains a query in the format described in the statement:

- $1\ x$;
- $2\ x\ p\ t$ ($1 \le p, t \le 2000$);
- $3\ x$;
- $4\ x\ p$ ($1 \le p \le 2000$).

Additional constraints on the input data:

- x in each query does not exceed the current number of stores (that is, 1 plus the number of type 1 queries);
- query type 3 is not applied to a store that has no types of ice cream;
- there is at least one query of type 4.

Output

For each query of type 4, output a single integer — for store x, find the maximum total tastiness of a subset of types of ice cream that are sold there, such that the total price does not exceed p (each type can be used in the subset no more than once).

Standard Input	Standard Output
12	4
2 1 5 7	11
2 1 3 4	0
4 1 4	11
4 1 8	17
4 1 2	

1 1	4
2 2 4 10	17
4 1 9	
4 2 9	
3 1	
4 1 9	
4 2 9	