


THÀNH PHỐ HỒ CHÍ MINH – NĂM 2022

MỤC LỤC

BÀI 1: KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU, CON TRỎ VÀ XỬ LÝ NGOẠI LỆ	Error! Bookmark not defined.
BÀI 2: CÁC GIẢI THUẬT TÌM KIẾM VÀ SẮP XẾP	Error! Bookmark not defined.
BÀI 3. MẢNG STRUCT & FILE.....	Error! Bookmark not defined.
BÀI 4: KỸ THUẬT XỬ LÝ MẢNG HAI CHIỀU.....	Error! Bookmark not defined.
BÀI 5. ÔN TẬP - KIỂM TRA LẦN 1	Error! Bookmark not defined.
BÀI 6. XỬ LÝ CHUỖI.....	Error! Bookmark not defined.
BÀI 7. KỸ THUẬT ĐỆ QUY	Error! Bookmark not defined.
BÀI 8. KỸ THUẬT ĐỆ QUY (tt).....	Error! Bookmark not defined.
BÀI 9. BÀI TẬP TỔNG HỢP	2
BÀI 10. ÔN TẬP - KIỂM TRA LẦN 2.....	6
PHỤ LỤC BÀI TẬP THỰC HÀNH NÂNG CAO.....	8

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p>BÀI 9.</p> <p>BÀI TẬP TỔNG HỢP</p>	
---	---	---

A. MỤC TIÊU:

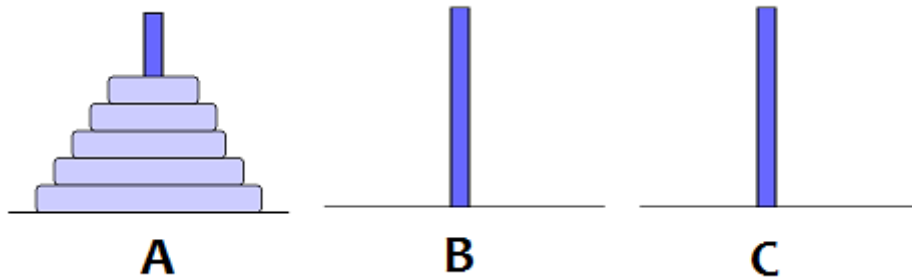
- Vận dụng các kiến thức xử lý ngoại lệ, xử lý mảng, xử lý chuỗi, kỹ thuật đệ quy để giải các bài toán thông dụng.
- Vận dụng các quy tắc để viết “clean code”

B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

Bài 1. Bài toán tháp Hà Nội: Có 3 chồng đĩa được đánh số A, B, C. Khởi đầu chồng đĩa A có n đĩa, được xếp sao cho đĩa lớn hơn luôn nằm dưới và B chồng đĩa còn lại chưa có đĩa nào. Hãy chuyển hết các đĩa từ chồng số 1 sang chồng số C, mỗi lần chỉ chuyển 1 đĩa, được dùng chồng số B làm trung gian, trong quá trình vận chuyển phải đảm bảo đĩa lớn hơn luôn nằm dưới.



Hàm Tower(n, colA, colB, colC)

Nếu n = 1 thì

Chuyển thẳng 1 đĩa (duy nhất) từ colA sang colC

Ngược lại

Tower(n – 1, colA, colC, colB)

// Chuyển n – 1 đĩa colA sang colB (colC làm trung gian)

MoveDisk(colA, colC) //Chuyển đĩa n từ colA sang colC

Tower(n- 1, colB, colA, colC)

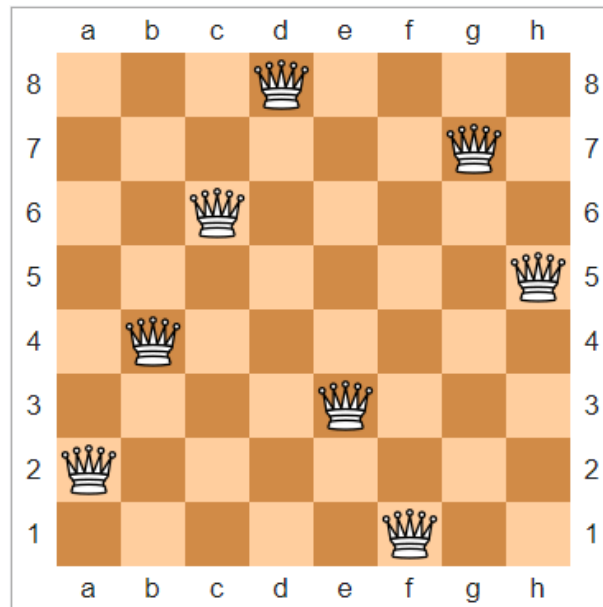
// Chuyển n -1 đĩa từ colB sang colC (colA trung gian)

Viết chương trình hiển thị thứ tự chuyển các đĩa theo phương pháp đệ quy.

Bài 2. Bài toán 8 quân hậu: đặt 8 quân hậu lên bàn cờ vua sao cho chúng không khống chế lẫn nhau. Viết chương trình tìm tất cả các lời giải.

Gợi ý:

Xếp tám quân hậu trên bàn cờ sao cho không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo. Bài toán tám quân hậu có thể tổng quát hóa thành bài toán đặt n quân hậu trên bàn cờ $n \times n$ (với $n \geq 4$).



Một trong 12 lời giải

Giải bài toán xếp hậu bằng đệ quy trong C/C++

- Để tiện trình bày ta dùng biến i để đánh dấu các hàng từ trên xuống (1 đến n). Dùng biến j để đánh dấu các cột từ trái sang phải (1 đến n);
- Các phần tử nằm trên cùng hàng có chỉ số hàng bằng nhau;
- Các phần tử nằm trên cùng cột có chỉ số cột bằng nhau;
- Để tiện cho việc in kết quả ra thì ta chỉ in ra chỉ số các cột tuần tự theo các hàng từ trên xuống.
- Điều kiện để đặt một quân hậu đúng chỗ là không có 2 trên cùng một cột (chỉ số cột khác nhau). Không có 2 quân hậu nào cùng ở trên một đường chéo.

Ý tưởng:

- Đầu tiên ta đặt quân hậu thứ nhất vào các cột trên hàng 1 (có n cách đặt).
- Thử đặt quân hậu 2 vào từng cột ở hàng 2 sao cho không bị quân hậu 1 khống chế. Với mỗi vị trí của quân hậu này ta lại thử đặt quân hậu thứ ba vào các cột sao cho không bị các quân hậu trước khống chế.
- Sau khi đặt xong quân hậu thứ tám thì in ra một cách đặt.

Bài 3. Bài toán mã đi tuần. Trên bàn cờ quốc tế 8×8 (64 ô), hãy viết hàm đệ quy sao cho con mã đi qua 64 nước sao cho mỗi ô chỉ đi qua 1 lần.

Gợi ý:

- Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.
- Nếu một quân mã đi hết 64 vị trí và tại vị trí cuối cùng có thể di chuyển đến vị trí bắt đầu thông qua một nước cờ thì đó gọi là một hành trình đóng
- Có những hành trình, trong đó quân mã sau khi đi hết tất cả 64 ô của bàn cờ và từ ô cuối của hành trình không thể đi về ô xuất phát chỉ bằng một nước đi. Những hành trình như vậy được gọi là hành trình mở.
- Cách di chuyển của một quân mã

Nước đi của một quân mã giống hình chữ L và nó có thể di chuyển tất cả các hướng. Ở một vị trí thích hợp thì quân mã có thể di chuyển đến được 8 vị trí.

- Tại 1 ô bất kỳ trên bàn cờ, con mã có thể đi tối đa 8 nước quanh vị trí hiện tại.

8							
7			X		X		
6		X				X	
5				♞			
4		X				X	
3			X		X		
2							
1							
	1	2	3	4	5	6	7

- Tại vị trí hiện tại, ta chọn thử 1 bước đi kế tiếp
- Nếu không thành công thì lần ngược để chọn bước khác
- Nếu thành công thì ghi nhận bước đi này.

Hướng dẫn giải bài toán mã đi tuần:

❖ Xây dựng bước đi cho quân mã

Gọi x, y là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ của quân mã sẽ như sau: $|x| + |y| = 3$ (với $x, y > 0$).

Khi đó ở một vị trí bất kỳ quân mã có 8 đường có thể di chuyển, chưa xét đến bước đi đó có hợp lệ hay không.

Các bước đi đó là: $(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1), (2, 1)$

Để đơn giản ta sẽ tạo hai mảng $X[]$, $Y[]$ để chứa các giá trị trên, với mỗi $X[i]$, $Y[i]$ sẽ là một cách di chuyển của quân mã ($0 \leq i \leq 7$).

❖ Kiểm tra tính hợp lệ của bước đi


Ta sẽ dùng một mảng hai chiều $A[n*n]$ để lưu vị trí của từng ô trong bàn cờ. Tất cả mảng đều khởi tạo giá trị là 0 (quân mã chưa đi qua).

Gọi x , y là vị trí hiện tại của quân mã, thì vị trí tiếp theo mà quân mã đi sẽ có dạng $x + X[i]$, $y + Y[i]$. Một vị trí được gọi là hợp lệ thì sẽ thỏa mãn tính chất sau:

- $0 \leq x + X[i] \leq n-1$.
- $0 \leq y + Y[i] \leq n-1$.

Nếu bước đi đó là bước đi đúng thì ta sẽ lưu thứ tự của bước đi đó vào mảng $A[x + X[i], y + Y[i]]$.

--HẾT--

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p>BÀI 10.</p> <p>ÔN TẬP - KIỂM TRA LẦN 2</p>	
--	---	---

A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng hai cho các bài toán.
- Áp dụng các kỹ thuật cài đặt xử lý chuỗi và đệ quy.

B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

C. NỘI DUNG THỰC HÀNH

ÔN TẬP - ĐỀ KIỂM TRA MẪU:

KIỂM TRA LẦN 2

MÔN: TH KỸ THUẬT LẬP TRÌNH – THỜI GIAN: 120 phút

(SV không được sử dụng tài liệu)

Câu 1. Tạo chương trình theo cấu trúc (khai báo thư viện, khai báo cấu trúc, khai báo hàm con, hàm main, thân hàm con), có hàm hiển thị danh sách bài thực hiện được và cho người dùng lựa chọn từng bài cần thực hiện. Trong chương trình cần xử lý ngoại lệ nếu có.

(1 điểm)

Câu 2. Cho mảng 1 chiều a chứa số nguyên. Viết các hàm xử lý sau theo kỹ thuật đệ quy:

2.1. Tính tích các số chẵn trong a

(1 điểm)

2.2. Xuất các phần tử có chứa chữ số 4 trong a.

(1 điểm)

Câu 3. Viết các hàm sau bằng kỹ thuật đệ quy

3.1. Tính $S(n) = \frac{1}{1.2^1} + \frac{2}{2.3^2} + \frac{3}{3.4^3} + \dots + \frac{n}{n.(n+1)^n}$

(1 điểm)

Viết hàm theo 2 cách đệ quy và khử đệ quy.

3.2. Tính tổng các chữ số chẵn của số nguyên dương N.

(1 điểm)

Viết hàm theo 2 cách đệ quy và khử đệ quy.

3.2. Xuất các số Fibonacci lẻ thuộc đoạn $[m, n]$, biết rằng công thức tính số Fibonacci như sau:

(1 điểm)

$$Fibonacci(n) = \begin{cases} 1 & \text{với } n \leq 2 \\ Fibonacci(n-1) + Fibonacci(n-2) & \text{với } n > 2 \end{cases}$$

Ví dụ: Các số chẵn Fibonacci thuộc đoạn [10, 30] gồm: 13, 21

Câu 4. Nhà thuốc tây quản lý thông tin thuốc gồm:

- Mã thuốc (5 ký tự)
- Tên thuốc (20 ký tự)
- Nhà sản xuất (20 ký tự)
- Dạng thuốc (10 ký tự) có giá trị là: viên, nước.
- Đơn giá (số thực)
- Công dụng (50 ký tự)

4.1. Tạo và xuất mảng 1 chiều chứa danh sách thuốc. **(2 điểm)**

4.2. Sắp xếp danh sách thuốc tăng dần theo mã thuốc. **(1 điểm)**

4.3. Tìm thuốc có mã số bắt đầu bằng 3 ký tự “T01” theo giải thuật Binary search. **(1 điểm).**

--HẾT--

PHỤ LỤC

BÀI TẬP THỰC HÀNH NÂNG CAO

Problem 1. Double-ended Strings

<https://codeforces.com/problemset/problem/1506/C>

You are given the strings aa and bb , consisting of lowercase Latin letters. You can do any number of the following operations in any order:

- if $|a| > 0$ (the length of the string a is greater than zero), delete the first character of the string a , that is, replace a with $a_2a_3\dots a_n$;
- if $|a| > 0$, delete the last character of the string aa , that is, replace a with $a_1a_2\dots a_{n-1}$;
- if $|b| > 0$ (the length of the string b is greater than zero), delete the first character of the string b , that is, replace b with $b_2b_3\dots b_n$;
- if $|b| > 0$, delete the last character of the string b , that is, replace bb with $b_1b_2\dots b_{n-1}$.

Note that after each of the operations, the string aa or b may become empty.

For example, if $a = \text{"hello"}$ and $b = \text{"icpc"}$, then you can apply the following sequence of operations:

- delete the first character of the string $a \Rightarrow a = \text{"ello"}$ and $b = \text{"icpc"}$;
- delete the first character of the string $b \Rightarrow a = \text{"ello"}$ and $b = \text{"cpc"}$;
- delete the first character of the string $b \Rightarrow a = \text{"ello"}$ and $b = \text{"pc"}$;
- delete the last character of the string $a \Rightarrow a = \text{"ell"}$ and $b = \text{"pc"}$;
- delete the last character of the string $b \Rightarrow a = \text{"ell"}$ and $b = \text{"p"}$.

For the given strings aa and bb , find the minimum number of operations for which you can make the strings aa and bb equal. Note that empty strings are also equal.

Input

The first line contains a single integer t ($1 \leq t \leq 100$). Then t test cases follow.

The first line of each test case contains the string aa ($1 \leq |a| \leq 20$), consisting of lowercase Latin letters.

The second line of each test case contains the string bb ($1 \leq |b| \leq 20$), consisting of lowercase Latin letters.

Output

For each test case, output the minimum number of operations that can make the strings a and b equal.

Example

input	Copy
5 a a abcd bc hello codeforces hello helo dhjakjsnasjhfkasafasd adjsnasjhfksvdafdser	
output	Copy
0 2 13 3 20	

Problem 2. GCD Sum

<https://codeforces.com/problemset/problem/1498/A>

The *gcdSum* of a positive integer is the *gcd* of that integer with its sum of digits. Formally, $gcdSum(x) = gcd(x, \text{sum of digits of } x)$ for a positive integer x . $gcd(a, b)$ denotes the greatest common divisor of a and b — the largest integer d such that both integers a and b are divisible by d .

For example: $gcdSum(762) = gcd(762, 7 + 6 + 2) = gcd(762, 15) = 3$.

Given an integer n , find the smallest integer $x \geq n$ such that $gcdSum(x) > 1$.

Input

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

Then t lines follow, each containing a single integer n ($1 \leq n \leq 10^{18}$).

All test cases in one test are different.

Output

Output t lines, where the i -th line is a single integer containing the answer to the i -th test case.

Example

input	Copy
3 11 31 75	
output	Copy
12 33 75	

Note

Let us explain the three test cases in the sample.

Test case 1: $n = 11$:

$$\gcdSum(11) = \gcd(11, 1 + 1) = \gcd(11, 2) = 1.$$

$$\gcdSum(12) = \gcd(12, 1 + 2) = \gcd(12, 3) = 3.$$

So the smallest number ≥ 11 whose $\gcdSum > 1$ is 12.

Test case 2: $n = 31$:

$$\gcdSum(31) = \gcd(31, 3 + 1) = \gcd(31, 4) = 1.$$

$$\gcdSum(32) = \gcd(32, 3 + 2) = \gcd(32, 5) = 1.$$

$$\gcdSum(33) = \gcd(33, 3 + 3) = \gcd(33, 6) = 3.$$

So the smallest number ≥ 31 whose $\gcdSum > 1$ is 33.

Test case 3: $n = 75$:

$$\gcdSum(75) = \gcd(75, 7 + 5) = \gcd(75, 12) = 3.$$

The \gcdSum of 75 is already > 1 . Hence, it is the answer.

Problem 3. k-LCM (easy version)

<https://codeforces.com/problemset/problem/1497/C1>

It is the easy version of the problem. The only difference is that in this version $k = 3$.

You are given a positive integer n . Find k positive integers a_1, a_2, \dots, a_k , such that:

- $a_1 + a_2 + \dots + a_k = n$
- $LCM(a_1, a_2, \dots, a_k) \leq \frac{n}{2}$

Here LCM is the **least common multiple** of numbers a_1, a_2, \dots, a_k .

We can show that for given constraints the answer always exists.

Input

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

Output

For each test case print k positive integers a_1, a_2, \dots, a_k , for which all conditions are satisfied.

Example

input	Copy
3 3 3 8 3 14 3	
output	Copy
1 1 1 4 2 2 2 6 6	

Problem 4. Max and Mex

<https://codeforces.com/problemset/problem/1496/B>

You are given a multiset S initially consisting of n distinct non-negative integers. A multiset is a set, that can contain some elements multiple times.

You will perform the following operation k times:

- Add the element $\lceil \frac{a+b}{2} \rceil$ (rounded up) into S , where $a = \text{mex}(S)$ and $b = \max(S)$. If this number is already in the set, it is added again.

Here \max of a multiset denotes the maximum integer in the multiset, and mex of a multiset denotes the smallest non-negative integer that is not present in the multiset. For example:

- $\text{mex}(\{1, 4, 0, 2\}) = 3$;
- $\text{mex}(\{2, 5, 1\}) = 0$.

Your task is to calculate the number of **distinct** elements in S after k operations will be done.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n, k ($1 \leq n \leq 10^5, 0 \leq k \leq 10^9$) — the initial size of the multiset S and how many operations you need to perform.

The second line of each test case contains n **distinct** integers a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — the numbers in the initial multiset.

It is guaranteed that the sum of n over all test cases does not exceed 10^5 .

Output

For each test case, print the number of **distinct** elements in S after k operations will be done.

Example

input	Copy
5 4 1 0 1 3 4 3 1 0 1 4 3 0 0 1 4 3 2 0 1 2 3 2 1 2 3	
output	Copy
4 4 3 5 3	

Note

In the first test case, $S = \{0, 1, 3, 4\}$, $a = \text{mex}(S) = 2$, $b = \max(S) = 4$, $\lceil \frac{a+b}{2} \rceil = 3$. So 3 is added into S , and S becomes $\{0, 1, 3, 3, 4\}$. The answer is 4.

In the second test case, $S = \{0, 1, 4\}$, $a = \text{mex}(S) = 2$, $b = \max(S) = 4$, $\lceil \frac{a+b}{2} \rceil = 3$. So 3 is added into S , and S becomes $\{0, 1, 3, 4\}$. The answer is 4.

Problem 5. Three swimmers

<https://codeforces.com/problemset/problem/1492/A>

Three swimmers decided to organize a party in the swimming pool! At noon, they started to swim from the left side of the pool.

It takes the first swimmer exactly a minutes to swim across the entire pool and come back, exactly b minutes for the second swimmer and c minutes for the third. Hence, the first swimmer will be on the left side of the pool after $0, a, 2a, 3a, \dots$ minutes after the start time, the second one will be at $0, b, 2b, 3b, \dots$ minutes, and the third one will be on the left side of the pool after $0, c, 2c, 3c, \dots$ minutes.

You came to the left side of the pool exactly p minutes after they started swimming. Determine how long you have to wait before one of the swimmers arrives at the left side of the pool.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases. Next t lines contains test case descriptions, one per line.

Each line contains four integers p, a, b and c ($1 \leq p, a, b, c \leq 10^{18}$), time in minutes after the start, when you came to the pool and times in minutes it take the swimmers to cross the entire pool and come back.

Output

For each test case, output one integer — how long you have to wait (in minutes) before one of the swimmers arrives at the left side of the pool.

Example

input	Copy
4 9 5 4 8 2 6 10 9 10 2 5 10 10 9 9 9	
output	Copy
1 4 0 8	

Note

In the first test case, the first swimmer is on the left side in $0, 5, 10, 15, \dots$ minutes after the start time, the second swimmer is on the left side in $0, 4, 8, 12, \dots$ minutes after the start time, and the third swimmer is on the left side in $0, 8, 16, 24, \dots$ minutes after the start time. You arrived at the pool in 9 minutes after the start time and in a minute you will meet the first swimmer on the left side.

In the second test case, the first swimmer is on the left side in $0, 6, 12, 18, \dots$ minutes after the start time, the second swimmer is on the left side in $0, 10, 20, 30, \dots$ minutes after the start time, and the third swimmer is on the left side in $0, 9, 18, 27, \dots$ minutes after the start time. You arrived at the pool 2 minutes after the start time and after 4 minutes meet the first swimmer on the left side.

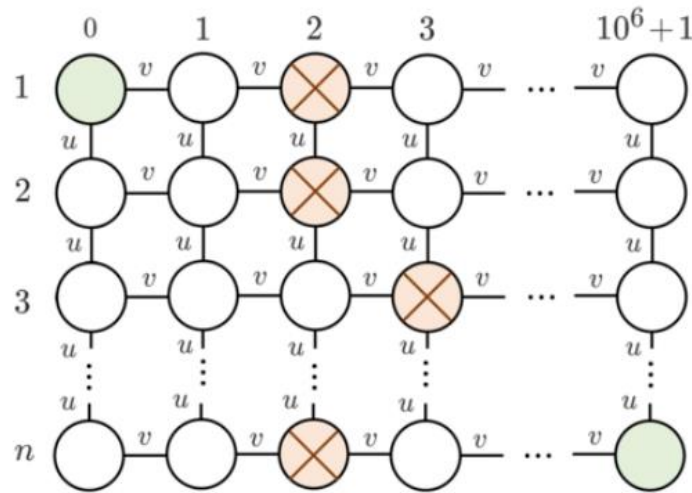
In the third test case, you came to the pool 10 minutes after the start time. At the same time, all three swimmers are on the left side. A rare stroke of luck!

In the fourth test case, all swimmers are located on the left side in $0, 9, 18, 27, \dots$ minutes after the start time. You arrived at the pool 10 minutes after the start time and after 8 minutes meet all three swimmers on the left side.

Problem 6. Minimal Cost

<https://codeforces.com/problemset/problem/1491/B>

There is a graph of n rows and $10^6 + 2$ columns, where rows are numbered from 1 to n and columns from 0 to $10^6 + 1$:



Let's denote the node in the row i and column j by (i, j) .

Initially for each i the i -th row has exactly one obstacle — at node (i, a_i) . You want to move some obstacles so that you can reach node $(n, 10^6 + 1)$ from node $(1, 0)$ by moving through edges of this graph (you can't pass through obstacles). Moving one obstacle to an adjacent by edge free node costs u or v coins, as below:

- If there is an obstacle in the node (i, j) , you can use u coins to move it to $(i - 1, j)$ or $(i + 1, j)$, if such node exists and if there is no obstacle in that node currently.
- If there is an obstacle in the node (i, j) , you can use v coins to move it to $(i, j - 1)$ or $(i, j + 1)$, if such node exists and if there is no obstacle in that node currently.
- Note that you **can't move obstacles outside the grid**. For example, you can't move an obstacle from $(1, 1)$ to $(0, 1)$.

Refer to the picture above for a better understanding.

Now you need to calculate the minimal number of coins you need to spend to be able to reach node $(n, 10^6 + 1)$ from node $(1, 0)$ by moving through edges of this graph without passing through obstacles.

Input

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

The first line of each test case contains three integers n , u and v ($2 \leq n \leq 100$, $1 \leq u, v \leq 10^9$) — the number of rows in the graph and the numbers of coins needed to move vertically and horizontally respectively.

The second line of each test case contains n integers a_1, a_2, \dots, a_n ($1 \leq a_i \leq 10^6$) — where a_i represents that the obstacle in the i -th row is in node (i, a_i) .

It's guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^4$.

Output

For each test case, output a single integer — the minimal number of coins you need to spend to be able to reach node $(n, 10^6 + 1)$ from node $(1, 0)$ by moving through edges of this graph without passing through obstacles.

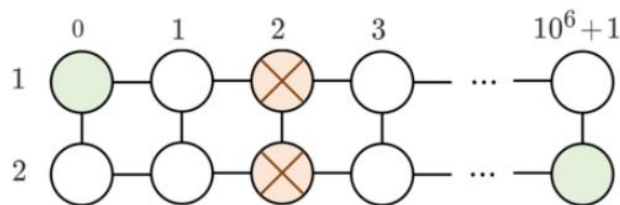
It can be shown that under the constraints of the problem there is always a way to make such a trip possible.

Example

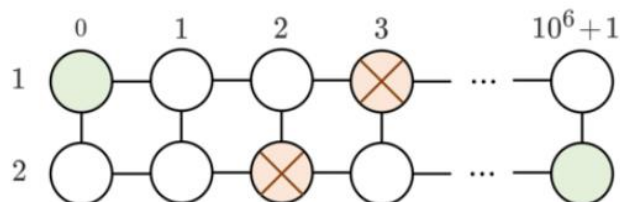
input	Copy
3	
2 3 4	
2 2	
2 3 4	
3 2	
2 4 3	
3 2	
output	Copy
7	
3	
3	

Note

In the first sample, two obstacles are at $(1, 2)$ and $(2, 2)$. You can move the obstacle on $(2, 2)$ to $(2, 3)$, then to $(1, 3)$. The total cost is $u + v = 7$ coins.



In the second sample, two obstacles are at $(1, 3)$ and $(2, 2)$. You can move the obstacle on $(1, 3)$ to $(2, 3)$. The cost is $u = 3$ coins.



Problem 7. Pythagorean Triples

<https://codeforces.com/problemset/problem/1487/D>

A Pythagorean triple is a triple of integer numbers (a, b, c) such that it is possible to form a right triangle with the lengths of the first cathetus, the second cathetus and the hypotenuse equal to a , b and c , respectively. An example of the Pythagorean triple is $(3, 4, 5)$.

Vasya studies the properties of right triangles, and he uses a formula that determines if some triple of integers is Pythagorean. Unfortunately, he has forgotten the exact formula; he remembers only that the formula was some equation with squares. So, he came up with the following formula: $c = a^2 - b$.

Obviously, this is not the right formula to check if a triple of numbers is Pythagorean. But, to Vasya's surprise, it actually worked on the triple $(3, 4, 5)$: $5 = 3^2 - 4$, so, according to Vasya's formula, it is a Pythagorean triple.

When Vasya found the right formula (and understood that his formula is wrong), he wondered: how many are there triples of integers (a, b, c) with $1 \leq a \leq b \leq c \leq n$ such that they are Pythagorean both according to his formula and the real definition? He asked you to count these triples.

Input

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

Each test case consists of one line containing one integer n ($1 \leq n \leq 10^9$).

Output

For each test case, print one integer — the number of triples of integers (a, b, c) with $1 \leq a \leq b \leq c \leq n$ such that they are Pythagorean according both to the real definition and to the formula Vasya came up with.

Example

input	Copy
3 3 6 9	
output	Copy
0 1 1	

Note

The only Pythagorean triple satisfying $c = a^2 - b$ with $1 \leq a \leq b \leq c \leq 9$ is $(3, 4, 5)$; that's why the answer for $n = 3$ is 0, and the answer for $n = 6$ (and for $n = 9$) is 1.

Problem 8. Guessing the Greatest (easy version)

<https://codeforces.com/problemset/problem/1486/C1>

The only difference between the easy and the hard version is the limit to the number of queries.

This is an interactive problem.

There is an array a of n different numbers. In one query you can ask the position of the second maximum element in a subsegment $a[l..r]$. Find the position of the maximum element in the array in no more than 40 queries.

A subsegment $a[l..r]$ is all the elements a_l, a_{l+1}, \dots, a_r . After asking this subsegment you will be given the position of the second maximum from this subsegment in the whole array.

Input

The first line contains a single integer n ($2 \leq n \leq 10^5$) — the number of elements in the array.

Interaction

You can ask queries by printing "? l r" ($1 \leq l < r \leq n$). The answer is the index of the second maximum of all elements a_l, a_{l+1}, \dots, a_r . Array a is fixed beforehand and can't be changed in time of interaction.

You can output the answer by printing "! p", where p is the index of the maximum element in the array.

You can ask no more than 40 queries. Printing the answer doesn't count as a query.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages

Hacks

To make a hack, use the following test format.

In the first line output a single integer n ($2 \leq n \leq 10^5$). In the second line output a permutation of n integers 1 to n . The position of n in the permutation is the position of the maximum

Example

input	Copy
5	
3	
4	
output	Copy
? 1 5	
? 4 5	
! 1	

Note

In the sample suppose a is $[5, 1, 4, 2, 3]$. So after asking the $[1..5]$ subsegment 4 is second to max value, and it's position is 3. After asking the $[4..5]$ subsegment 2 is second to max value and it's position in the whole array is 4.

Note that there are other arrays a that would produce the same interaction, and the answer for them might be different. Example output is given in purpose of understanding the interaction.

Problem 9. The Robot

<https://codeforces.com/problemset/problem/1468/K>

There is a robot on a checkered field that is endless in all directions. Initially, the robot is located in the cell with coordinates $(0, 0)$. He will execute commands which are described by a string of capital Latin letters 'L', 'R', 'D', 'U'. When a command is executed, the robot simply moves in the corresponding direction:

- 'L': one cell to the left (the x -coordinate of the current cell decreases by 1);
- 'R': one cell to the right (the x -coordinate of the current cell is increased by 1);
- 'D': one cell down (the y -coordinate of the current cell decreases by 1);
- 'U': one cell up (the y -coordinate of the current cell is increased by 1).

Your task is to put an obstacle in one cell of the field so that after executing the commands, the robot will return to the original cell of its path $(0, 0)$. Of course, an obstacle cannot be placed in the starting cell $(0, 0)$. It is guaranteed that if the obstacle is not placed, then the robot will not return to the starting cell.

An obstacle affects the movement of the robot in the following way: if it tries to go in a certain direction, and there is an obstacle, then it simply remains in place (the obstacle also remains, that is, it does not disappear).

Find any such cell of the field (other than $(0, 0)$) that if you put an obstacle there, the robot will return to the cell $(0, 0)$ after the execution of all commands. If there is no solution, then report it.

Input

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

Each test case consists of a single line containing s — the sequence of commands, which are uppercase Latin letters 'L', 'R', 'D', 'U' only. The length of s is between 1 and 5000, inclusive. Additional constraint on s : executing this sequence of commands leads the robot to some cell other than $(0, 0)$, if there are no obstacles.

The sum of lengths of all s in a test doesn't exceed 5000.

Output

For each test case print a single line:

- if there is a solution, print two integers x and y ($-10^9 \leq x, y \leq 10^9$) such that an obstacle in (x, y) will force the robot to return back to the cell $(0, 0)$;
- otherwise, print two zeroes (i. e. 0 0).

Example

input	Copy
4 L RUUDL LLUU DDUUUUU	
output	Copy
-1 0 1 2 0 0 0 1	

Problem 10. Prison Break

<https://codeforces.com/problemset/problem/1415/A>

There is a prison that can be represented as a rectangular matrix with n rows and m columns. Therefore, there are $n \cdot m$ prison cells. There are also $n \cdot m$ prisoners, one in each prison cell. Let's denote the cell in the i -th row and the j -th column as (i, j) .

There's a secret tunnel in the cell (r, c) , that the prisoners will use to escape! However, to avoid the risk of getting caught, they will escape at night.

Before the night, every prisoner is in his own cell. When night comes, they can start moving to adjacent cells. Formally, in one second, a prisoner located in cell (i, j) can move to cells $(i - 1, j)$, $(i + 1, j)$, $(i, j - 1)$, or $(i, j + 1)$, as long as the target cell is inside the prison. They can also choose to stay in cell (i, j) .

The prisoners want to know the minimum number of seconds needed so that every prisoner can arrive to cell (r, c) if they move optimally. Note that there can be any number of prisoners in the same cell at the same time.

Input

The first line contains an integer t ($1 \leq t \leq 10^4$), the number of test cases.

Each of the next t lines contains four space-separated integers n, m, r, c ($1 \leq r \leq n \leq 10^9, 1 \leq c \leq m \leq 10^9$).

Output

Print t lines, the answers for each test case.

Example

input	Copy
3 10 10 1 1 3 5 2 4 10 2 5 1	
output	Copy
18 4 6	

--HẾT--