Winter Camp Contest 2020

NCTU PCCA

ID	Problem Name	Time Limit
A	Ascend the Alps	1 sec
B	Blocking Buildings	1 sec
C	Capoo's Christmas	1 sec
D	Dope Design	1 sec
E	Eerie Echo	1 sec
F	Flawlessly Fortified	1 sec
G	Gap Gambolling	1 sec
H	Hard to Handle	1 sec

NCTU PCCA Winter Camp Contest 2020
Contest Information

Contest Rules

Contestants who fail to abide the rules will be disqualified.

1. A team shall consist of at most three (3) contestants.
2. No machine-readable materials (e.g., source codes, templates, etc.) are allowed. However, paper-based materials, such as textbooks, dictionaries, printed notes, etc., are allowed.
3. Contestants are only allowed to contact their teammates during the contest. Contestants shall not discuss with their coach and other teams.
4. Contestants shall only access the internet for downloading the problem description, submitting source codes, requesting problem clarification and checking the scoreboard. Any other type of internet access is prohibited.
5. A team shall not simultaneously use more than one computer to write programs during the contest. Contestant shall not use any other type of electronic devices, except extra monitors and printers.
6. All malicious actions interfering the contest are prohibited.

Scoring & Ranking

1. Contestants who are disqualified do not count towards the rankings.
2. Teams must submit their solutions via DOMjudge. Each submission will be given an ID and a timestamp of its submission time. It is guaranteed that a submission with greater ID is submitted later. The judge system will only response to submissions that is submitted within the contest duration (180 minutes). The response to each run must be one of the following:
   - `Correct`: The judge accepts your code.
   - `Compiler-Error`: Your code cannot be successfully compiled.
   - `TimeLimit`: Your program consumes too much time.
   - `Run-Error`: Your program terminates with an non-zero return code, which often means your program is terminated by the operating system.
   - `Wrong-Answer`: The judge rejects the output of your program.
   - `No-Output`: Your program does not generate any output.

3. Teams are ranked according to the most problems solved. Teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the accepted run plus 20 penalty minutes for every rejected run for that problem regardless of submittal time. There is no time consumed for a problem that is not solved.

4. If multiple teams have the same number of problems solved and total time, then the winner goes to the team which first submitted their last accepted solution.

Almost blank page

Problem A
Ascend the Alps

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

Capoo often climbs mountains as a hobby. In his opinion, the best part about mountain climbing is that he can view the gorgeous landscape from the top of a mountain. One day, he and his friends decided to start an adventure on the Alps, one of the most famous mountain ranges on Earth.

The Alps can be considered as a "polyline" on the plane; it consists of n vertices $A_1, A_2, \ldots, A_n$ and $n-1$ segments $\overline{A_1A_2}, \overline{A_2A_3}, \ldots, \overline{A_{n-1}A_n}$ connecting them. For two points P and Q on it, we say that they are visible from each other if the entire line segment $\overline{PQ}$ (excluding its endpoints) is strictly on top of the polyline. Note that for each point P, P is visible from itself.

Since Capoo's main goal is to admire the beautiful scenery of the mountains, he wants to know how high he can see from each vertex A_i $(1 \le i \le n)$, i.e. the maximum y-coordinate among that of all points that are visible from A_i.

Input Format

The first line contains an integer n. Then n lines follow, the i^{th} of which contains two integers x_i y_i denoting the coordinates of the i^{th} vertex: $A_i = (x_i, y_i)$.

Output Format

Output n space-separated integers s_1 s_2 $\ldots$ s_n denoting the answer to each vertex, i.e. s_i is the maximum y-coordinate among that of all points that are visible from A_i.

Technical Specification

- $2 \le n \le 2 \times 10^5$
- $0 \le x_i, y_i \le 10^9$
- $x_1 < x_2 < \cdots < x_n$

Sample Input 1

```
5
1 2
2 3
4 4
6 1
7 5
```

Sample Output 1

```
2 3 5 1 5
```

Almost blank page

Problem B
Blocking Buildings

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

Capoo owns a street on which N buildings lie in a row. The buildings are numbered 1 to n from left to right, and the i^{th} one has height A_i.

While taller buildings give more profit, they've got a major downside: the city's view may get obstructed by them. Since Capoo likes to take afternoon walks on his street, he does not wish that the buildings are too tall, or they might block the awesome scenery of the city.

Capoo has planned his walks for the next M days. On the j^{th} day, he will start from the L_j-th building, walk right to the R_j-th one, and admire the sublime views while walking. The "satisfiability" of the walk he will gain on day j is given by the number of buildings between the L_j-th and the R_j-th one (inclusive) with height not greater than K_j.

Given the plans, Capoo wants to know the satisfiability for each day in advance. However, since he is not good at solving these type of problems (he always falls asleep during algorithm classes), he wants you to write a program to help him.

Input Format

The first line consists of two integers N and M, the amount of buildings and days Capoo has a plan for. On the second line there are N numbers $A_1\ A_2\ \ldots\ A_N$ describing the buildings' height. Then M lines follow: on the j^{th} line there are three integers $L_j\ R_j\ K_j$ describing the walk on day j.

Output Format

Output M lines, on the j^{th} line print an integer: the satisfiability of the walk on day j.

Technical Specification

- $1 \le M, N, A_i, K_i \le 2 \times 10^5$
- $1 \le L_i \le R_i \le N$

Sample Input 1

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

Sample Output 1

```
5
3
2
```

Almost blank page

Problem C
Capoo's Christmas

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

It's Christmas time in Capoo's world! To celebrate, he and his friends planted N Christmas trees in a row, which are numbered 1 to N from left to right. To make it more festive, they decorated each tree so that the i^{th} one has a shininess value a_i.

After New Year, Capoo realized that the trees has become quite redundant, so he decided to remove M of them, one at a time: the i^{th} tree he will remove has index b_i. After some removals, the remaining trees form several *connected segments*, each of which consists of one or more trees with contiguous indices. For example, the trees with indices 3, 4, and 5 form a connected segment of length 3.

Each tree has a *gorgeousness* value, defined as its shininess multiplied by the length of the longest connected segment it belongs to. Capoo wants to know, after each removal, the maximum gorgeousness value of the remaining trees.

Input Format

The first line contains two integers N M, denoting the number of planted trees and the number of trees that are to be removed. One the second line there are N space-separated integers $a_1\ a_2\ \ldots\ a_N$, the trees' shininess values. On the third line there are M space-separated integers $b_1\ b_2\ \ldots\ b_M$, the indices of trees in the order they are removed.

Output Format

Output M lines: on the i^{th} line, print the maximum gorgeousness value of the trees after the i^{th} removal.

Technical Specification

- $1 \leq M < N \leq 2 \times 10^5$
- $1 \leq a_i \leq 10^9$
- $1 \leq b_i \leq N$
- The indices b_i are pairwise distinct.

Sample Input 1

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

Sample Output 1

```
15
5
3
```

Note

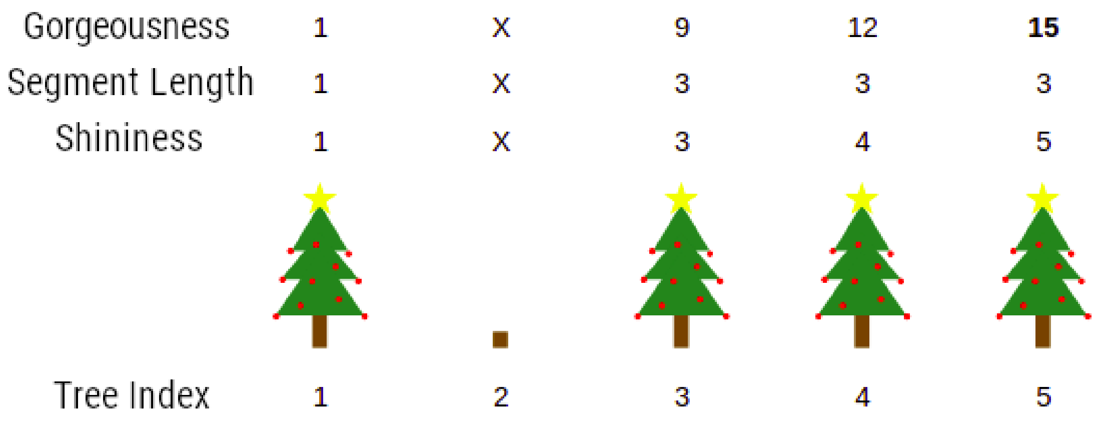

Figure 1: After removing the tree with index 2, the maximum gorgeousness becomes 15.

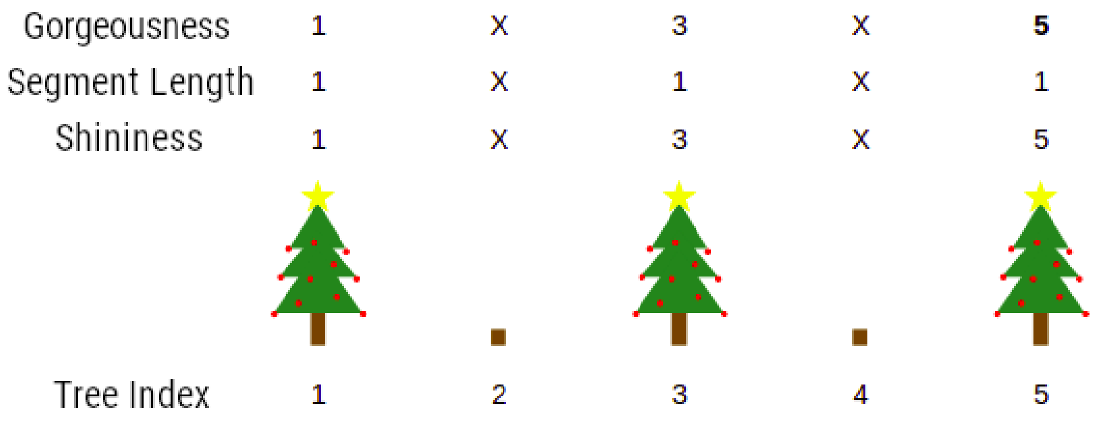

Figure 2: After removing the tree with index 4, the maximum gorgeousness becomes 5.

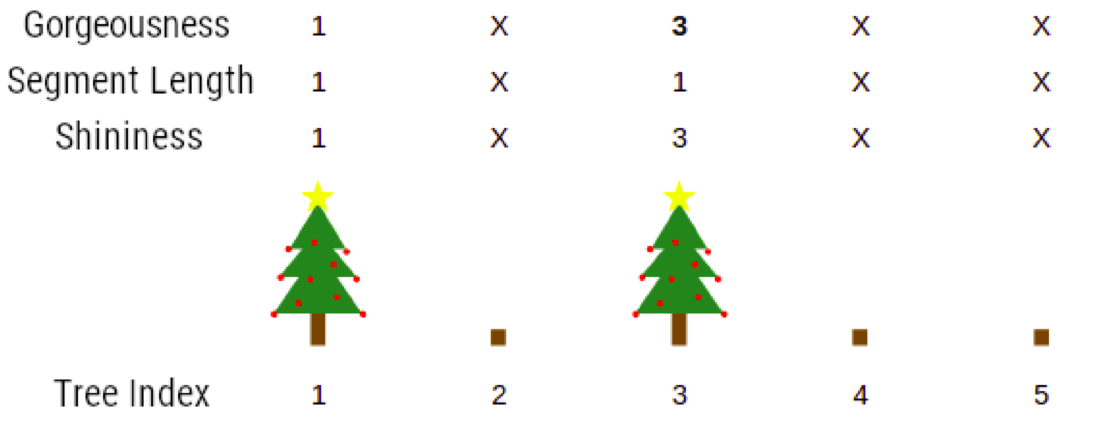

Figure 3: After removing the tree with index 5, the maximum gorgeousness becomes 3.

Problem D
Dope Design

Time limit: 1 second
Memory limit: 512 megabytes

Problem Description

Capoo really enjoys playing with building blocks recently, and has been studying various methods to improve the structures he built with those blocks. Of all structures, the one he likes the most is the modern *deque design*. A deque design consists of N stacks of building blocks in a row, numbered 1 through N from left to right. The i^{th} stack is built from a_i blocks of height 1, i.e. it has height a_i where a_i is a positive integer. Moreover, Capoo calls a deque design *dope* if each of the following holds:

1. There exists an unique index k $(1 < k < n)$ such that the k^{th} stack has maximum height; all other stacks are strictly shorter than it.
2. The stacks' heights are non-decreasing before the k^{th} one, i.e. $a_1 \leq a_2 \leq \cdots \leq a_{k-1}$.
3. The stacks' heights are non-increasing after the k^{th} one, i.e. $a_{k+1} \geq a_{k+2} \geq \cdots \geq a_n$.

Capoo wants to know, if the tallest stack is not taller than M, how many different dope deque designs with N stacks can he build? Two designs differ if there exists a stack that has different height in the first design and in the second one.

Input Format

The input contains two integers N and M, the number of stacks and their height limit.

Output Format

Output one integer, the number of different dope designs Capoo can create. Since the answer can be large, output it modulo 998244353.

Technical Specification

- $3 \leq N \leq 10^5$
- $2 \leq M \leq 200$

Sample Input 1

```
3 3
```

Sample Output 1

```
5
```

Sample Input 2

```
7 4
```

Sample Output 2

```
481
```

Sample Input 3

```
100000 200
```

Sample Output 3

```
323684606
```

Almost blank page

Problem E
Eerie Echo

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

There was an eerie melody echoing in the valley. Enchanted by it, Froggo couldn't help but kept jumping on lily pads in a lake.

After a while, Froggo found that each lily pad has an unique integer label written on it, and that from the one labeled x he could only jump to the one labeled ♪(x).

♪(x) can be written as the difference $\sharp(x) - \flat(x)$, where both of them depend on D, the number of digits in x. The two functions are defined as:

- $\sharp(x)$: The digits of x sorted by non-ascending order.
- $\flat(x)$: The digits of x sorted by non-descending order.

All labels have the same number of digits, and it's possible that a label have leading zeros.

Froggo knows for sure that he will eventually get stuck jumping on lily pads in an endless cycle, but he is not sure about the length of it. Given the number of digits D and the label of Froggo's starting point N, find the length of the cycle he will get into.

Input Format

The input consists of multiple test cases. The first line contains a single integer T, the number of test cases. For each test case, there are 2 integers D and N in one line.

Output Format

For each test case, output an integer in one line — the number of lily pads in the cycle.

Technical Specification

- $1 \leq T \leq 10^4$
- $1 \leq D \leq 10$
- $1 \leq N < 10^D$
- The integer N will be padded to D digits with leading zeros.
- It is guaranteed that Froggo will get into a cycle shortly.

Sample Input 1

```
2
4 0020
8 64356804
```

Sample Output 1

```
1
3
```

Note

- Test case 1

  1. $♪(0020) = \sharp(0020) - \flat(0020) = 2000 - 0002 = 1998$
  2. $♪(1998) = \sharp(1998) - \flat(1998) = 9981 - 1899 = 8082$
  3. $♪(8082) = \sharp(8082) - \flat(8082) = 8820 - 0288 = 8532$
  4. $♪(8532) = \sharp(8532) - \flat(8532) = 8532 - 2358 = 6174$
  5. $♪(6174) = \sharp(6174) - \flat(6174) = 7641 - 1467 = 6174$

  $0020 \to 1998 \to 8082 \to 8532 \to 6174 \to 6174$
  The cycle: 6174

- Test case 2

  1. $♪(64356804) = \sharp(64356804) - \flat(64356804) = 86654430 - 03445668 = 83208762$
  2. $♪(83208762) = \sharp(83208762) - \flat(83208762) = 88763220 - 02236788 = 86526432$
  3. $♪(86526432) = \sharp(86526432) - \flat(86526432) = 86654322 - 22345668 = 64308654$
  4. $♪(64308654) = \sharp(64308654) - \flat(64308654) = 86654430 - 03445668 = 83208762$

  $64356804 \to 83208762 \to 86526432 \to 64308654 \to 83208762$
  The cycle: $83208762 \to 86526432 \to 64308654$

Problem F
Flawlessly Fortified

Time limit: 1 second
Memory limit: 512 megabytes

Problem Description

The frogs' empire consists of N^2 villages which form a N by N square grid, and between any two neighboring villages there is a bidirectional road connecting them. Traveling via the roads is the only way to move between villages.

One day, a mysterious virus outbreak occurred at $(1, 1)$ — the top-left village. The virus is highly contagious, so when a frog travels to other villages, the virus may get spread there as well. To prevent getting infected, the king of frogs decides to take some measures to stop the virus from spreading. He can do any of the following:

- Choose the two villages at (i, j) and $(i + 1, j)$, and close down the road between them. This can make the villagers unhappy, so he has to pay the villages $A_{i,j}$ coins and $A_{i+1,j}$ coins respectively.
- Choose the two villages at (i, j) and $(i, j + 1)$, and close down the road between them. Similarly, he has to pay the villages $A_{i,j}$ coins and $A_{i,j+1}$ coins respectively.
- Choose the village at (i, j) and completely seal it off — this will prevent any frog from entering or leaving it. The cost to do this is $B_{i,j}$ coins.

The king's main goal is to avoid virus from spreading to (N, N), the village in which he lives. He wants to perform some actions described above, so that it's impossible for a frog to travel from $(1, 1)$ to (N, N). Since he really wants to save coins (so he can buy more games), please help him determine the minimum cost to do so.

Input Format

The first line contains an integer N.
Then N lines follow, each containing N space-separated integers. The j^{th} number on the i^{th} line is $A_{i,j}$.
Then N more lines follow, each containing N space-separated integers. The j^{th} number on the i^{th} line is $B_{i,j}$.

The test cases may contain very large inputs. For C++ users, it is advised to add `ios_base::sync_with_stdio(0), cin.tie(0), cout.tie(0);` before your first `cin` or use `scanf` / `printf` instead, so your program doesn't consume too much time while reading the input.

Output Format

Output one integer, the minimum cost required to satisfy the king.

Technical Specification

- $2 \le N \le 1000$
- $0 \le A_{i,j}, B_{i,j} \le 10^6$

Sample Input 1

```
4
1 99 99 99
1 1 99 99
99 99 1 1
99 99 1 99
99 99 99 99
99 99 99 1
99 1 99 99
99 99 99 99
```

Sample Output 1

```
10
```

Note

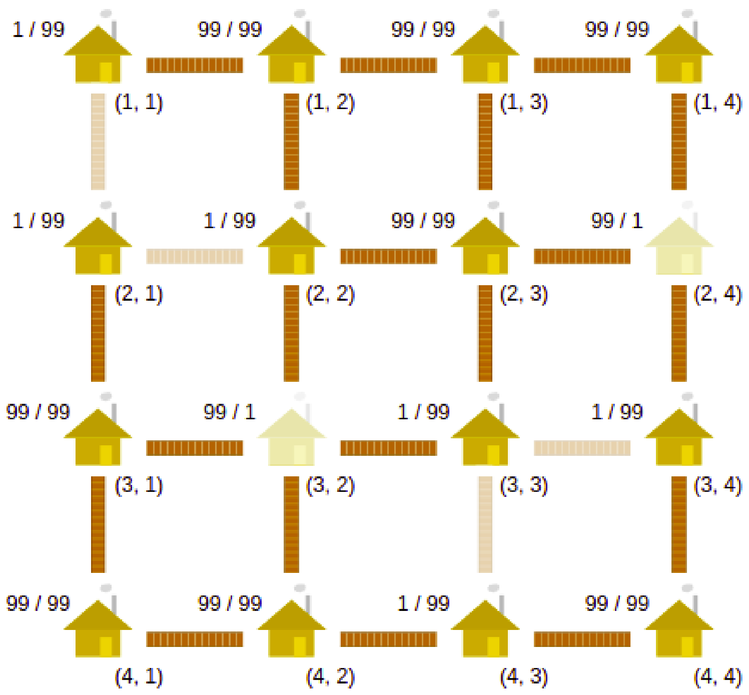

Figure 4: Village layout in the sample test. The A and B values are written on the top left of each village. If one closes down the dimmed village and roads, they could achieve the minimum cost of 10.

Problem G
Gap Gambolling

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

There are N frogs who play in the river every day. Their favorite thing to do is hopping around on the rocks in the river, and occasionally they play the game of "ribbit-hop". The rules of this game is as follows:

- Select N rocks and place them along a straight line, then number them 1 to N from left to right.
- Choose positive integers $a_1, a_2, \dots, a_N$ and write a_i on the i^{th} rock.
- Initially, there is a frog on each rock.
- On each move, every frog will do a jump: each frog on the j^{th} rock will jump right to the $(j + a_j)$-th one. If $j + a_j > N$, then it jumps onto the riverbank instead. After a move, it is possible that a rock has multiple frogs on it.

The frogs will do K moves in total. After all moves are done, please determine the amount of frogs that are on the riverbank.

Input Format

The first line contains two integers N and K, the amount of rocks/frogs and the number of total moves. The second line contains N space-separated integers $a_1\ a_2\ \dots\ a_N$ denoting the numbers written on rocks.

Output Format

Output one integer, the number of frogs that will be on the riverbank after K moves.

Technical Specification

- $1 \le N, K \le 2 \times 10^5$
- $1 \le a_i \le N$

Sample Input 1

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

Sample Output 1

```
3
```

Note

In the sample test, the initial frogs on rocks 1, 4 and 5 will jump onto the riverbank.

Almost blank page

Problem H
Hard to Handle

Time limit: 1 second

Memory limit: 512 megabytes

Problem Description

As a well-known fact, frogs are really fussy animals and are often hard to handle.

Frogs love even numbers so much that, they will only have their dinner if there are a **positive** even number of dishes with weights in even numbers on the table.

Please help the chef check if a given set of dishes are acceptable by frogs. If not, you can try to make it so by secretly removing one dish.

Input Format

The first line contains a single integer n, the number of dishes. The second line contains n space-separated integers $a_1\ a_2\ \ldots\ a_n$, the weights of dishes.

Output Format

If the dishes are acceptable, print `YES` on the first line. On the second line, print the weights that are even in non-descending order, separated by spaces.

If you have to remove a dish, print `OuO` on the first line. On the second line, print the weight of the dish that should be removed. If there are multiple answers, you should output the minimum one.

Otherwise, print `NO`.

Technical Specification

- $1 \le n \le 10^5$
- $1 \le a_i \le 10^{18}$

Sample Input 1

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

Sample Output 1

```
YES
2 4
```

Sample Input 2

```
5
6 8 4 3 7
```

Sample Output 2

```
OuO
4
```

Sample Input 3

```
5
1 5 3 7 11
```

Sample Output 3

```
NO
```