NOI 2024 Qualification

24 February 2024

Tasks	Time Limit	Memory Limit
Tourist	1 second	1GB
Party	1 second	1GB
Photo	1 second	1GB
Park	1 second	1GB
Explosives	1 second	1GB

Have Fun!

Task 1: `Tourist`

You are a tourist who wishes to explore a city for N days. Your itinerary has already been planned out, and you will need to take $a[i]$ train journeys on day i ($1 \leq i \leq N$).

In order to take the train, you need to buy train tickets. There are two kinds of train tickets sold:

- A **single trip ticket** costs x dollars and allows you to take a single trip.
- A **day pass** costs y dollars and allows you unlimited travel for the day.

Find the minimum cost you need to spend on the train tickets.

Input format

Your program must read from standard input.

The first line of input contains exactly 3 integers, n, x, y.

The next line contains n space-separated integers $a[1], a[2], \ldots, a[n]$.

Output format

Your program must print to standard output.

The output should contain one integer, the minimum cost (in dollars) that you need to spend on train tickets.

The output should contain only a single integer. Do not print any additional text such as `Enter a number` or `The answer is`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 1000$

- $1 \leq x \leq 1000$
- $1 \leq y \leq 10^6$
- $1 \leq a[i] \leq 10^4$ (for all $1 \leq a[i] \leq n$)

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	100	No additional constraints

Sample Testcase 1

Input	Output
3 4 9 2 3 2	25

Sample Testcase 1 Explanation

You are travelling for a total of 3 days, a single trip ticket costs 4 dollars, and a day pass costs 9 dollars.

On day 1, you need to take 2 trips. You can buy 2 single trip tickets for 8 dollars.

On day 2, you need to take 3 trips. You can buy a day pass for 9 dollars.

On day 3, you need to take 2 trips. You can buy 2 single trip tickets for 8 dollars.

In total, you spend 25 dollars. This is the minimum possible cost you need to spend.

Task 2: `Party`

James has n friends and wants to host a party. He wants to invite zero or more friends and he knows that they will come if they get invited. Each friend gets $a[i]$ happiness from attending the party. Note that some friends may not want to go and have negative $a[i]$.

He needs to seat the friends he invites but due to social distancing measures, no two friends can sit beside each other. He only has n seats, help him figure out who he should invite to maximize the total happiness they get!

Input format

Your program must read from standard input.

The first line contains a single integer n

The second line contains n integers $a[i]$ representing the happiness each friend can get.

Output format

Your program must print to standard output.

The output should contain one integer, the maximum total happiness of friends who get invited. Do not print any additional text such as `'Enter a number'` or `'The answer is'`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 200\,000$
- $-10^9 \leq a[i] \leq 10^9$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	49	$n \leq 3$
2	38	$n \leq 1000$
3	13	No additional constraints

Sample Testcase 1

This testcase is valid for subtasks 2 and 3.

Input	Output
5 3 2 -1 4 5	12

Sample Testcase 1 Explanation

James has 5 seats. He can invite friends 1, 4 and 5 and seat them as such, leaving an empty seat between them.

1		4		5

This brings his friends a total happiness of $a_1 + a_4 + a_5 = 3 + 4 + 5 = 12$.

Sample Testcase 2

This testcase is valid for all subtasks.

Input	Output
1 10	10

Sample Testcase 2 Explanation

James can invite the one friend and seat them in the one seat gaining a total happiness of 10.

Sample Testcase 3

This testcase is valid for subtasks 2 and 3.

Input	Output
6 1 -3 2 10 -4 9	21

Sample Testcase 3 Explanation

James can invite friends 3, 4, and 6 and sit them as such, leaving an empty seat between them.

3		6		4	

This brings his friends a total happiness of $a_3 + a_4 + a_6 = 2 + 10 + 9 = 21$.

Task 3: `School Photo`

Zane is the principal of NOI school. NOI school has n classes, and each class has s students. Student j in class i has height $h[i][j]$.

Zane wants to select 1 student from each class to take a school photo. To make the photo look nicer, Zane wants to select students such that the height difference between the tallest student and shortest student selected is as small as possible.

Input format

Your program must read from standard input.

The first line of input contains exactly 2 integers, n, the number of classes, and s, the number of students in each class.

The next n lines contain information about the classes. Line $i + 1$ contains s integers $h[i][j]$, representing the height of the students in class i.

Output format

Your program must print to standard output.

The output should contain 1 integer, the minimum height difference possible.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $2 \leq n \leq 1\,000$
- $1 \leq s \leq 1\,000$
- $1 \leq h[i][j] \leq 10^9$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	11	$n = 2$
2	22	$n, s \leq 100$
3	9	$n, s \leq 250$
4	33	$n, s \leq 500$
5	25	No additional restrictions

Sample Testcase 1

This testcase is valid for all subtasks.

Input	Output
2 3 2 1 8 5 4 7	1

Sample Testcase 1 Explanation

There are 2 classes in NOI school with 3 students each. Class 1 has students with height 2, 1 and 8 respectively, while class 2 has students with height 5, 4 and 7 respectively.

To minimise the height difference, Zane can choose the student with height 8 from class 1, and the student with height 7 from class 2. This makes the height difference equal to $8 - 7 = 1$, which is the minimum possible.

Sample Testcase 2

This testcase is valid for subtasks 2 to 5.

Input	Output
3 3 3 1 4 2 7 18 9 8 10	4

Sample Testcase 2 Explanation

Zane can choose the student with height 4 from class 1, the student with height 7 from class 2 and the student with height 8 from class 3. This makes the height difference equal to $8 - 4 = 4$, which is the minimum possible.

Task 4: `Amusement Park`

People usually go to amusement parks in groups. But what happens when, during their turn to board a ride, they are told that there are not enough seats for all of them? Some groups are willing to split, whereas some aren't.

Stuart the Snail manages the queue in front of a popular attraction. Groups of people queue up to board. During the boarding period, Stuart first receives the number of seats that can be filled. Then, he approaches each group in the queue in order, starting from the group in front. He tells the group the number of seats remaining and asks if they would like to board. The group makes a decision like this:

- If there are enough seats for the whole group, the group will board together.
- Otherwise, if the group is willing to split, they send enough people to fill up the remaining seats while the rest of the group stays in their position in the queue.
- Otherwise, the entire group stays in their position in the queue together.

It is possible that not all available seats are filled by the end of this process. Regardless, the people that manage to board will enjoy the attraction and leave without rejoining the queue.

Unfortunately, this process is very time-consuming for a snail like Stuart as he needs to move along the queue repeatedly. Please help to write a program that keeps track of the status of the queue and quickly determines which groups should split up if necessary and board the attraction. Specifically, it should support the following three operations:

- *join*: A group joins the back of the queue.
  - You are given an integer s, the size of the group.
  - You are also given an integer w, which is either 0 or 1. If $w = 0$, then the group is unwilling to split. Otherwise, if $w = 1$, then the group is willing to split. You may assume that a group's willingness to split never changes.
  - Suppose this is the i-th *join* operation, then this group is assigned the ID i, where i starts counting from 1.
- *leave*: A group leaves the queue without boarding the attraction.
  - You are given an integer i, the ID of the group leaving the queue.
  - It is guaranteed that this group is in the queue. If the group is willing to split, it is possible that some of its members have already boarded the attraction and left.

- *board*: Allow some people to board the ride according to the rules.
  - You are given an integer b, the maximum number of people that can board during this period.
  - You should decide how many people from each group should board. For additional information, refer to the *Output format* section.

Input format

Your program must read from standard input.

Note that some values in the input may not fit in a 32-bit integer.

The first line of input contains a single integer q, representing the total number of *join*, *leave* and *board* operations.

The next q lines each contain either two or three integers, following one of the three formats listed below:

- $1\ \ s\ \ w$ describing a *join* operation;
- $2\ \ i$ describing a *leave* operation;
- $3\ \ b$ describing a *board* operation.

Output format

Your program must print to standard output.

For *join* and *leave* operations, **do not output anything.**

For each *board* operation, let k be the number of groups where at least one person will board the attraction. You should output $k + 1$ lines as described below.

- The first line should contain only the integer k.
- If $k > 0$, each of the remaining k lines should contain two integers. The first is the group ID, whereas the second is the number of people from that group which will board the attraction. **You should output the k lines in order of increasing group ID.** (If $k = 0$, you may ignore this part.)

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq q \leq 200\,000$
- For all *join* operations, $1 \leq s \leq 200\,000$ and $0 \leq w \leq 1$
- For all *leave* operations, the group with ID i is in the queue at the time of the operation
- For all *board* operations, $1 \leq b \leq 10^{12}$
- There is at least one *board* operation

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	12	$q \leq 1000$
2	7	$s = 1$, $w = 0$, there are no *leave* operations
3	20	$s \leq 10$, $w = 0$, there are no *leave* operations
4	16	$s \leq 10$, there are no *leave* operations
5	10	$s \leq 10$
6	35	No additional restrictions

Sample Testcase 1

This testcase is valid for subtasks 1, 5 and 6.

Input	Output
7 1 2 0 1 6 0 1 6 1 3 5 2 2 1 3 0 3 123456789012	2 1 2 3 3 2 3 3 4 3

Sample Testcase 1 Explanation

After the first 3 operations, the groups from the front to the back of the queue are as follows:

- ID 1, size 2, not willing to split
- ID 2, size 6, not willing to split
- ID 3, size 6, willing to split

For the 4th operation, seats are assigned as follows:

- Initially, there are 5 seats available. The 2 people in the group with ID 1 can board.
- There are 3 seats remaining, which is not enough for the group with ID 2. They are not willing to split, so none of them board.
- There are still 3 seats remaining, which is not enough for the group with ID 3, but they are willing to split, so 3 of them board.

There are $k = 2$ groups where at least one person boards the attraction, corresponding to group IDs 1 and 3. The output for this operation is

```
2
1 2
3 3
```

Now the groups from the front to the back of the queue are as follows:

- ID 2, size 6, not willing to split
- ID 3, size 3, willing to split

In the 5th and 6th operations, the group with ID 2 leaves while a new group with ID 4 joins. The queue looks like this:

- ID 3, size 3, willing to split
- ID 4, size 3, not willing to split

In the 7th operation, there are clearly more than enough seats for everyone in the queue. There are $k = 2$ groups where at least one person boards the attraction, corresponding to group IDs 3 and 4. The output for this operation is

```
2
3 3
4 3
```

Sample Testcase 2

This testcase is valid for all subtasks.

Input	Output
5 1 1 0 1 1 0 1 1 0 3 2 1 1 0	2 1 1 2 1

Sample Testcase 3

This testcase is valid for subtasks 1 and 6.

Input	Output
4 1 19 1 3 10 3 10 3 10	1 1 10 1 1 9 0

Task 5: `Explosives`

You are an explosive handler in charge of transporting explosives. There are n factories and n mines located on a straight line. The factories produce explosives and the mines require explosives.

Each factory produces 1 unit of explosives, and each mine requires 1 unit of explosives. The i-th factory ($1 \leq i \leq n$) is located at position $a[i]$ and the j-th mine ($1 \leq j \leq n$) is located at position $b[j]$. The factories and mines all have different locations (i.e. $a[1 \ldots n], b[1 \ldots n]$ are all distinct).

As the explosive handler, you drive a truck which can carry at most c units of explosives. You start at location 0 and your truck contains no explosives initially. Your aim is to transport all the explosives from factories to mines.

More specifically, you can execute the following operations:

1. `pickup(x)`: Drive from your current location to the factory located at x, and pickup 1 unit of explosive from this factory. You are only allowed to do this if
   - $x = a[i]$ (for some $1 \leq i \leq n$)
   - Your truck contains at most $c - 1$ units of explosives.

   This operation increases the amount of explosives on your truck by 1.
2. `offload(x)`: Drive from your current location to the mine located at x, and offload 1 unit of explosive to this mine. You are only allowed to do this if
   - $x = b[j]$ (for some $1 \leq j \leq n$)
   - Your truck contains at least 1 unit of explosives.

   This operation decreases the amount of explosives on your truck by 1.

You should pickup exactly 1 unit of explosive at every factory, and offload exactly 1 unit of explosive at every mine.

Whenever there are explosives on the truck, you need to pay a safety officer to watch the explosives for you.

In particular, if you drive from x to y carrying at least 1 unit of explosives, then you need to pay the officer a cost of $|x - y|$, which does not depend on the amount of explosives on your truck. Note that you do not need to pay this cost when there are no explosives on your truck.

Construct a sequence of operations that minimizes this cost.

Input format

Your program must read from standard input.

The first line of input contains two integers, n and c.

The second line will contain n integers $a[1 \ldots n]$.

The third line will contain n integers $b[1 \ldots n]$.

Output format

Your program must print to standard output.

The first line should contain the minimum cost required. The second line should contain $2n$ integers, indicating the positions of all the factories and mines you visit, in order.

For example, if your operations are `pickup(3), offload(5), pickup(6), offload(2)` in that order, the second line should contain `3 5 6 2`.

Subtasks

For all testcases, the input will satisfy the following bounds:

- $1 \leq n \leq 1000$
- $1 \leq c \leq 1000$
- $1 \leq a[i], b[i] \leq 10000$ (for all $1 \leq i \leq n$)
- $a[1 \ldots n], b[1 \ldots n]$ are all distinct (i.e. all factories and mines have distinct locations).

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
0	0	Sample Testcases
1	16	$c = 1$
2	22	$a[i] \leq 5000, b[i] > 5000$ (for all $1 \leq i \leq n$)
3	62	No additional restrictions

For each subtask, you will get 50% of points if you output the correct minimum cost for all test cases in the subtask

Sample Testcase 1

Input	Output

`3 2`
`12 14 4`
`9 5 8`
	`7`
	`4 5 14 12 9 8`

Sample Testcase 1 Explanation

There are 3 factories, located at positions 12, 14 and 4. There are 3 mines, located at positions 9, 5 and 8.

A possible sequence of operations is `pickup(4), offload(5), pickup(14), pickup(12), offload(9), offload(8)`. The cost of these operations can be calculated as follows:

- Initially, there are no explosives on the truck. You do not need to pay to move from 0 to 4.
- Once we pickup at position 4, there is now 1 unit of explosive on your truck. The operation `offload(5)` costs $|5 - 4| = 1$. The truck is now empty (i.e. contains no explosives).
- `pickup(14)` does not cost you anything as the truck is now empty.
- `pickup(12)` costs $|14 - 12| = 2$ as the truck contains 1 unit of explosives.
- `offload(9)` costs $|12 - 9| = 3$
- `offload(8)` costs $|9 - 8| = 1$
- The cost you pay is therefore $1 + 2 + 3 + 1 = 7$.

If your output had just been:

```
7
```

Then you would get 50% of the points. Note that you are **not allowed** to print anything after the cost and it has to be correct for all test cases in the subtask.