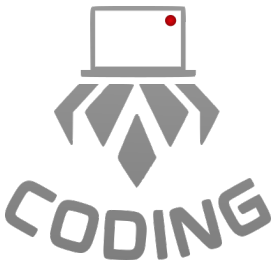
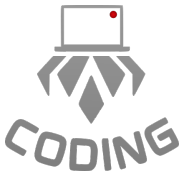


AOI 2022 Day 1

Anderson Secondary School - Robotics Club Coding Group

22 August 2022



Instructions

Some of these questions have very large input and output sizes. Some require both the use of C++ and its fast I/O. Here is a template for fast I/O that you may use.

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(0); // Warning: this will turn off all stdio functions
    // so printf and scanf and cannot be used anymore
    cin.tie(0); // Warning: this will mess with the timing of your output,
    // so do not worry if it does not output anything before you are done with
    // your input

    // All your code MUST go below these 2 lines or fast I/O will not be activated
    // Use cin and cout from now onward for input and output
}
```

Scoring

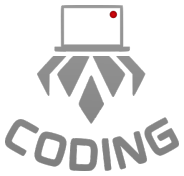
Scoring will be based on (in order of priority from top to bottom):

1. Total marks
2. Difficulty of question and subtasks solved
3. Number of questions fully solved

Time taken to solve will not be taken into account.

Rules

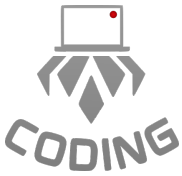
1. Strictly no communication is allowed aside from clarifications through Google Classroom.
2. You may NOT search online for any information.
3. You may use any of your pre-prepared Google Documents as you wish if approved by us. (Approach us beforehand)
4. You may also wish to use the AOI and C++ references we provided in Google Classroom.



Recommended advice to follow:

1. Try ALL questions, even if you think they are way out of your league.
2. Never give up. There's always a subtask or two or ten to grab.
3. You are allowed (and recommended) to submit the same code for different subtasks.
4. Read the questions carefully.

Happy programming! :)



Task A: Equality

Shor the duck and his K-1 friends are hungry, and you accidentally went to the park with N pieces of food!

You have no clue why you brought along food for the ducks, perhaps they brainwashed you into doing so.

Now, they implore you to distribute the food you have as equally as possible. You can only give integer amounts of food to each duck, and you **MUST GIVE ALL YOUR FOOD TO THEM OR FACE THE CONSEQUENCES.**

Furthermore, each duck can get jealous of other ducks if they receive more food than them. The jealousy of each duck i is defined as

$$J_i = (D_{max} - D_i)^2$$

Where J_i is the jealousy of duck i, D_{max} is the most number of food pieces any one duck receives, and D_i is the number of food pieces duck i receives.

Your distribution must then minimise the sum of jealousy values of all ducks, $\sum(J_i)$.

Input Format

Your program must read from standard input.

The input consists of only two numbers, N and K, separated by spaces on a single line.

Output Format

Your program must print to standard output.

The output should consist of only one number, the minimum sum of jealousy levels that can be achieved.

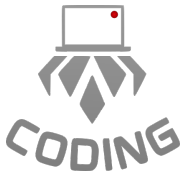
Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 512MB. For all test cases, the input will satisfy the following bounds:

- $0 \leq N \leq 10^{15}$
- $1 \leq K \leq 10^{14}$

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

Subtask	Marks	Additional Constraints
1	76	$0 \leq N \leq 10^2, 1 \leq K \leq 10$
2	24	No Additional Restrictions



Sample Input 1

```
5 2
```

Sample Output 1

```
1
```

Sample Testcase 1 Explanation

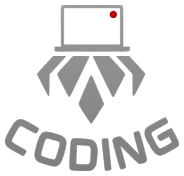
If the ducks are given 2 and 3 pieces of food respectively, the jealousy values are 0 and 1, and minimum sum is 1. It can be proven that this is the minimum sum. Note that there may be multiple distributions that give the minimum sum.

Sample Input 2

```
341432 10
```

Sample Output 2

```
8
```



Task C: Toilet Disruptions

Take all timings in 24 hour format HH:MM.

Doo the Dog only has N days before he has to submit his final PhD thesis about how dogs are superior to cats. However, he has just gotten food poisoning, oh no!

As a result of the food poisoning, he has to go to the toilet very frequently throughout all N days. Namely, on the i-th day, there are T_i different time periods where he needs to go to the toilet. Don't ask why he knows the time periods he needs to go to the toilet, he can foresee the future.

On the i-th time period, on the j-th period, he starts going to the toilet at $S_{i,j}$ minutes from 00:00 and is done at $E_{i,j}$ minutes from 00:00. For example, $S_0 = 120, E_0 = 200$ means that Doo the dog goes to the toilet and can't do anything from 02:00 to 03:20 inclusive. Also, he does not sleep at all cuz he is a programmer.

Since his thesis is due very soon, he would like to work on it for as long as possible. However, Doo the dog has a severe case of ADHD and gets distracted easily when he goes to the toilet. He will also get very tired after working on his thesis. So he can only work on his thesis in 1 continuous time period where he does not need to go to the toilet in the time period. (Note that this time period can span across multiple days)

Given the number of days he has left, N, the time periods in each day where he has to go to the toilet, T_i, and the time intervals he has to be in the toilet, $S_{i,j}$ and $E_{i,j}$. Help Doo the dog the number of minutes which he can work on his thesis for the longest.

Input Format

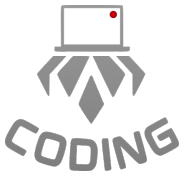
Your program must read from standard input.

The first line contains 1 integer, N, the number of days before his thesis is due.
For N instances, the first line contains T_i, the number of periods he has to go to the toilet on the i-th day.
For T_i instances, the first line contains 2 integers, $S_{i,j}$ and $E_{i,j}$, the starting time and ending time he is in the toilet. (You may ask for clarification on input cuz yes this is very confusing).

N
T_0
S_0 E_0
S_1 E_1
S_2 E_2
...
S_{T_0} E_{T_0}
T_1
...
T_{N-1}
$S_{T_{N-1}}$ $E_{T_{N-1}}$

Output Format

Your program must print to standard output.



The output should contain 1 integer, the longest time Doo the dog can work on his thesis for. (The longest duration he does not need to go to the toilet across all N days).

Implementation Note

You MUST use c++ and ios_base::sync_with_stdio(0); cin.tie(0); or else it will TLE. The AMOS committee has not found a solution capable of solving this question without it.

Subtasks

The maximum execution time on each instance is 1.0s, and the maximum memory usage on each instance is 512MB. For all test cases, the input will satisfy the following bounds:

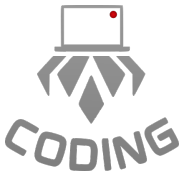
- $1 \leq N \leq 2 \cdot 10^5$
- $1 \leq T_i \leq 10$
- $0 \leq S_{i,j} \leq E_{i,j} < 1,440$
- All time periods where he goes to the toilet will never overlap or go over any day

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

Subtask	**Marks**	**Additional Constraints**
1	10	$N = 1$
2	10	$T_i = 1$
3	80	No Additional Restrictions

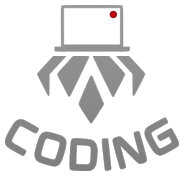
Sample Testcase 1

Input	Output
3 1 0 1439 2 120 600 800 801 5 20 40 150 350 351 780 790 800 1000 1400	658



Sample Testcase 1 Explanation

The first time period which he is free is 0 to 119, followed by 601 to 799, 802 to 19 (Next day), 41 to 149, 781 to 789, 801 to 999, 1401 to 1439. The longest time period here is the 4-th one, 802 to 19 (Next day), as the number of minutes he is free is (1439-801) + (20) = 658 minutes. This is higher than all other periods. (1439-801) is the number of minutes left in day 2 from minute 802 to 1439 inclusive. (20) is the number of minutes left in day 1 from minute 0 to 19 inclusive, because 20 is occupied by toilet.



Task E: Art of War

LCJLY the general is having a war with Ramsey Institution. As a general, his aim is to win the battle with minimal firepower while having minimal losses.

In the battle, LCJLY has a total of X tanks. According to Sun Tzu's Art of War, tanks have a strength of 3. His enemy (Ramsey Institution) has their forces positioned in N different cities and each city has a total strength of A_i. As LCJLY the generals need to capture all of the cities to consider winning the war he needs to find out how to deploy his troop most efficiently with minimal reinforcements needed. LCJLY's superior AMOS has ordered him to fight the cities in sequence, starting from the city 1 to city N, capturing every city possible. If LCJLY's army's power is more than or equal to the current city's power he is at, he must attack it as AMOS has instructed. He will stop attacking once he the city's power is more than the power of his army that is leftover. Help LCJLY find the most number of cities he can attack with the optimal usage of tanks.

In order to capture a city, you need to deploy units that have equal to or larger strength than the defensive strength of the city. Once you have defeated a city, the number of tanks will decrease with the number of tanks needed to use.

Input Format

Your program must read from standard input.

The first line of input consists of 1 integers X (X is the number of tanks)
The second line of input is an integer N (number of cities needed to be captured)
The third line of input consists of N integer A_i

Output Format

Your program must print to standard output.

The output should contain the index of the very last city LCJLY can destroy if he destroys every city he encounters. If he cannot destroy any, print "You are a loser".

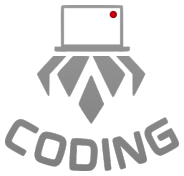
Implementation Note

You might need to use C++ and the fast I/O template given in the first page to avoid TLE.

Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 512MB. For all test cases, the input will satisfy the following bounds:

- $0 \leq N \leq 10^6$
- $0 \leq A_i \leq 10^5$
- $0 \leq X \leq 10^9$



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

Subtask	Marks	Additional Constraints
1	12	$0 \leq A_i \leq 1$
2	24	A_i is divisible by 3
3	24	A_i is less than or equal to 3
4	40	No Additional Restrictions

Sample Input 1

```
1
2
4 0
```

Sample Output 1

```
You are a loser
```

Sample Testcase 1 Explanation

The first city cannot be captured with 1 tank, so he stops attacking and isn't able to capture any cities. (Note: City 2's power is less than LCJLY's power but he cannot capture it as he must capture in order from 1 to N).

Sample Input 2

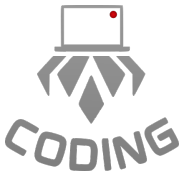
```
4
4
1 10 5 7
```

Sample Output 2

```
1
```

Sample Testcase 2 Explanation

The first city can be taken with 1 tank, leaving behind 3 tanks. The second city cannot be captured with the remaining 3 tanks, so LCJLY stops attacking even if the next cities 3 and 4 can be captured.



Task G: Duck Snacks

After many years of grinding for duck snacks, Shor the duck has established a huge industry of linear Duck Snack farms with a field size of N. Many of the workers can't be bothered to traverse the whole gigantic field to deposit the Duck Snacks, so instead they leave them in K piles around the field. Each pile of duck snacks, the i-th pile being located at X_i, and containing Q_i Duck Snacks, both of which are guaranteed to be integers. (The location of the left-most field is 1 and the right-most is N).

However, there are geese who want to sabotage the duck's food supply lines, and will steal or destroy the Duck Snacks. Shor the Duck knows of their plans, and has created an ingenious system to protect the Duck Snacks from their heinous plans. The protection system has a protection range of B and can be placed anywhere in the field, covering all integer locations X to X+B-1 inclusive, where $1 \leq X, X+B-1 \leq N$.

Shor the duck wants to know what is the maximum amount of duck snacks that can be protected with his protection system. Help him find the answer.

Input Format

Your program must read from standard input.

The first line contains 3 integers, N, K and B (The integers can fit into a 64-bit signed integer).
The second line contains K integers, denoting the positions of each pile, X_i.
The third line also contains K integers, denoting the amount of Duck Snacks in each pile, Q_i.

Output Format

Your program must print to standard output.

The output should contain 1 integer, the maximum amount of Duck Snacks that can be protected.

Subtasks

The maximum execution time on each instance is 1.0s, and the maximum memory usage on each instance is 512MB. For all test cases, the input will satisfy the following bounds:

- $1 \leq N, B \leq 10^{15}$
- $1 \leq K \leq 10^6$
- $K \leq N$
- All X_i are unique
- $1 \leq X_i \leq N$
- $1 \leq Q_i \leq 10^5$

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

Subtask	Marks	Additional Constraints
1	12	$B \geq N$
2	18	X_i is presorted by distance from the left end. (Increasing order) $1 \leq N \leq 2 \cdot 10^3$
3	21	X_i is presorted by distance from the left end. $1 \leq N \leq 10^6$
4	16	$1 \leq N \leq 10^6$
5	33	No Additional Constraints

Sample Testcase 1

Input	Output
5 5 5 1 2 3 4 5 8 9 6 5 34	62

Sample Testcase 1 Explanation

All five piles can be protected by selecting the range 1 to 5, so the answer is the sum of all Duck Snacks in all piles.

Sample Testcase 2

Input	Output
6 3 2 3 1 4 5 12 3	12

Sample Testcase 2 Explanation

We can protect the pile at position 1 by selecting the range of 1 to 2.
This protects 12 Duck Snacks in total, so our answer is 12 Duck Snacks.

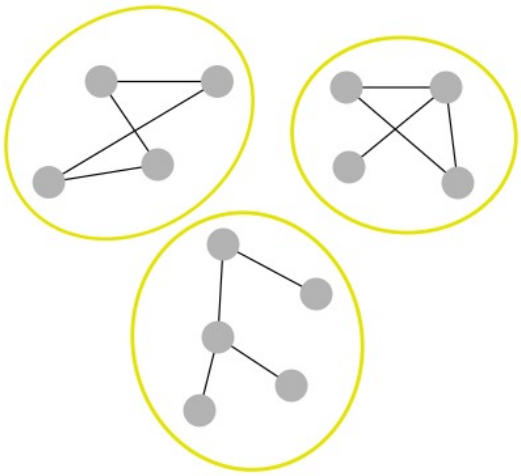
While we can protect two piles by selecting the range of 3 to 4, they only contain 8 Duck Snacks in total, and thus it is more optimal to choose the range previously mentioned instead.

Special Graph Theory Task: Nice Ducks

Shor the Duck is playing Santa this Christmas! He will be distributing presents to N Nice Ducks, labelled from 1 to N, as a reward for their good behaviour over the year. The house of each duck is represented by a node, and each node has a "Niceness Level". Namely, the i-th duck has a "Niceness level" of S_i. The higher the "Niceness Level", the Nicer the Duck has been over the year and hence the more deserving to receive a present from Shor the Duck.

However, there are unfortunately too many Nice Ducks this year, and Shor the Duck only has time to distribute the gifts to some Nice Ducks. More specifically, he only has time to distribute presents to the Neighbourhood with the highest total "Niceness Level".

A neighbourhood is defined as a disjoint subgraph of nodes, where every node is connected directly or indirectly to every other node in the subgraph with edges. There are E edges, the i-th edge connecting house A_i and B_i.



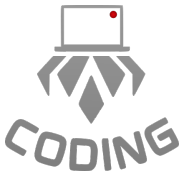
The gray circles represent nodes, the house of a duck, the black lines represent edges which connect a house to other houses. There are 3 different disjoint subgraphs shown in the picture represented as the yellow circles.

Find the highest "Niceness Level" of a neighbourhood so that Shor the Duck can distribute presents to the Nicest Neighbourhood!

Input Format

Your program must read from standard input.

2 integers, N (total number of nodes) and E (total number of bidirectional edges).
A row of N integers will then follow, containing the array A, representing the Niceness level of each house.
Finally, E rows of two integers, A_i and B_i, will follow, representing an edge between node A_i and B_i.



Output Format

Your program must print to standard output.

Output 1 integer, the maximum "Niceness Level" of a neighbourhood (Note that it is possible for this to be negative)

Implementation Note

You might need to use c++ and the fast io template to avoid TLE. Refer to 1st page to get fast io template.

Subtasks

The maximum execution time on each instance is 2.0s, and the maximum memory usage on each instance is 512MB. For all test cases, the input will satisfy the following bounds:

- $1 \leq N \leq 10^6$
- $0 \leq E \leq 10^6$
- $-10^6 \leq S_i \leq 10^6$
- $1 \leq A_i, B_i \leq N$
- A_i and B_i are never equal

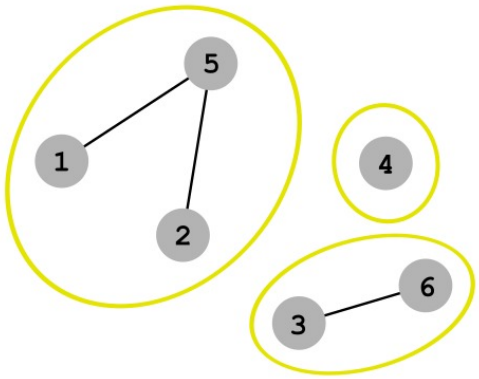
Your program will be tested on input instances that satisfy the following restrictions:

Subtask	**Marks**	**Additional Constraints**
1	10	$S_i = 0$
2	10	$E = 0$
3	40	$S_i = 1$
4	50	$0 \leq E \leq 10^3$, $1 \leq N \leq 10^3$
5	90	No Additional Constraints

Sample Testcase 1

Input	Output
6 3 100 200 50 1 -100 200 1 5 2 5 3 6	250

Sample Testcase 1 Explanation



The yellow circles are the neighborhoods, there are 3. The neighborhood (1, 2, 5) has a niceness of 100 + 200 + (-100) = 200. The neighborhood (4) has a niceness of 1. The neighborhood (3, 6) has a niceness of 50 + 200 = 250. Therefore, the highest niceness level of a neighborhood is 250.