AIO.

2024

AUSTRALIAN INFORMATICS OLYMPIAD

AUSTRALIAN MATHS TRUST

DATE

Thursday 29 August

DURATION

3 hours

- There are six problems. All problems are of equal value (100 marks).
- The problems are sorted by approximate difficulty, but you may attempt them in any order.
- The contest runs for 3 hours.
- You may submit once per minute per problem. You may make an unlimited number of submissions.
- Submit solutions as you write them, do not wait until the last minute to submit them.

Contest website: https://aio.edu.au/

STOP!

Do not open the paper until you are instructed to do so.

PROBLEM 1

Javelin

Input file: `javelin.txt`
Output file: `javelout.txt`

Time and memory limits: 1 second, 1 GB

N students participated in the javelin throwing event at your school's latest athletics meet. The students took turns throwing the javelin as far as possible, and the ith student threw it D_i metres. The winner was the student who threw it the furthest. Fortunately, every student threw the javelin a different distance and so there were no ties.

You want to see how exciting the competition was by counting how many different students were leading the competition at some point. Specifically, a student is considered a *current leader* if they threw the javelin further than everyone who came before them. How many different current leaders were there during the competition?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $2 \leq N \leq 200\,000$.
- $1 \leq D_i \leq 1\,000\,000$ for all i.
- All D_i are different. That is, $D_i \neq D_j$ for all $i \neq j$.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (40 marks), $N = 2$.
- For Subtask 2 (40 marks), $N \leq 1000$.
- For Subtask 3 (20 marks), no special rules apply.

Input

Your program must read input from the file `javelin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `javelin.txt` follows a specific format:

- The 1st line contains the integer N.
- The 2nd line contains N integers, describing how far each student threw the javelin. The ith of these is D_i.

Output

Your program must write a single integer to the file `javelout.txt`: the number of current leaders during the competition.

Sample input 1

```
3
1 2 3
```

Sample output 1

```
3
```

Sample input 2

```
6
5 2 3 8 1 7
```

Sample output 2

```
2
```

Sample input 3

```
2
2 1
```

Sample output 3

```
1
```

Explanation

- In the 1st sample case, all three students were current leaders because they threw further than everyone before them.
- In the 2nd sample case, the 1st student (who threw it 5 metres) and the 4th student (who threw it 8 metres) were the only two current leaders.
- In the 3rd sample case, the 1st student was the only current leader.

PROBLEM 2

Subbookkeeper

Input file: `bookin.txt`
Output file: `bookout.txt`

Time and memory limits: 1 second, 1 GB

Rebecca loves words, and has devised a system to calculate which words are her favourite. She gives each word a *score*, which is the number of times that two adjacent letters are the same:

- `REBECCA` has a score of 1, because it has adjacent `C`s.
- `ANNABELLA` has a score of 2, because it has adjacent `N`s and adjacent `L`s.
- `HELLLLLO` has a score of 4, because `LLLLL` has four sets of adjacent `L`s.
- `HAHAHA` has a score of 0, because it has no adjacent letters that are the same.

Rebecca has recently started a job as the subbookkeeper of the Woolloomooloo library, where she ranks 2nd only to the main bookkeeper Annabella. Part of her job involves inspecting books to see if any letters have rubbed off the pages. She must then write in the missing letters. Rebecca has decided to choose the replacement letters so that the score of the word is as large as possible. Rebecca's words are simply groups of letters. They don't have to be real words from the dictionary.

For example, imagine that she found the word `RE?EL`, where `?` represents a missing letter. If she chose `P`, then it would make the word `REPEL` with a score of 0. However, if she instead chose `E`, this would make `REEEL` with a score of 2.

Rebecca has found a word with exactly one missing letter. What is the largest score she can achieve by strategically choosing the replacement letter?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $3 \leq N \leq 200\,000$, where N is the number of letters in the word.
- The word consists of uppercase letters from `A` to `Z`, with exactly one `?`.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (30 marks), the `?` is the 1st letter in the word.
- For Subtask 2 (40 marks), the `?` is not the 1st nor the last letter in the word.
- For Subtask 3 (30 marks), no special rules apply.

Input

Your program must read input from the file `bookin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `bookin.txt` follows a specific format:

- The 1st line contains the integer N, which is the number of letters in the word.
- The 2nd line contains the word: exactly `N` characters, where one is `?` and the others are uppercase letters.

Output

Your program must write a single integer to the file `bookout.txt`: the largest score Rebecca can achieve by strategically choosing the replacement letter.

Sample input 1

```
5
RE?EL
```

Sample input 2

```
8
?ELLLLLO
```

Sample input 3

```
13
SUBBOO?KEEPER
```

Sample output 1

```
2
```

Sample output 2

```
5
```

Sample output 3

```
4
```

Explanation

- In the 1st sample case, the best word Rebecca can make is `REEEL` with a score of 2.
- In the 2nd sample case, the best word she can make is `EELLLLLO` with a score of 5 (1 from the `E`s and 4 from the `L`s).
- In the 3rd sample case, she can make either `SUBBOOKKEEPER` or `SUBBOOOKEEPER`, which both have a score of 4.

PROBLEM 3

Shopping Spree

Input file: `shopin.txt`
Output file: `shopout.txt`

Time and memory limits: 1 second, 1 GB

After winning the lottery you have made the (perhaps unwise) decision to buy every item at the local Pair Mart store. Pair Mart has an unusual policy where customers are only allowed to buy items in pairs. When buying a pair, you get two items for the cost of the more expensive one.

In addition, you have collected K Pair Mart coupons. Each coupon allows you to buy a pair of items for the cost of the cheaper one, rather than the more expensive one.

The store has N items and the ith item costs C_i dollars. You know that N is even and so you can buy every item in the store.

What is the minimum cost to buy all N items?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $2 \leq N \leq 200\,000$ and N is even.
- $0 \leq K \leq N/2$.
- $1 \leq C_i \leq 10\,000$ for all i.
- $C_i \leq C_{i+1}$. That is, the costs are given in sorted order.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (35 marks), $K = 0$ and $N \leq 1000$.
- For Subtask 2 (35 marks), $K = N/2$ and $N \leq 1000$.
- For Subtask 3 (30 marks), no special rules apply.

Input

Your program must read input from the file `shopin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `shopin.txt` follows a specific format:

- The 1st line of input contains the integers N and K.
- The 2nd line of input contains N integers describing the costs of the items. The ith of these is C_i.

Output

Your program must write a single integer to the file `shopout.txt`: the minimum cost (in dollars) to buy all the items.

Sample input 1

```
4 1
3 4 6 10
```

Sample output 1

```
9
```

Sample input 2

```
4 0
5 10 15 20
```

Sample output 2

```
30
```

Sample input 3

```
6 3
2 2 4 4 5 6
```

Sample output 3

```
8
```

Explanation

- In the 1st sample case, you can buy the 1st and 4th item using the coupon for 3 dollars. You can then buy the 2nd and 3rd item for 6 dollars. This comes to a total cost of 9 dollars, which is the lowest possible.
- In the 2nd sample case, you can buy the 1st and 2nd item together and the 3rd and 4th item together. This comes to a total cost of 30 dollars, which is the lowest possible.
- In the 3rd sample case, one way to achieve the lowest possible cost is to buy the 1st and 5th items for 2 dollars using a coupon, then buy the 2nd and 6th items for 2 dollars using a coupon, and then finally buy the 3rd and 4th item for 4 dollars. The last pair will cost 4 dollars no matter whether or not you use the coupon. This comes to a total cost of 8 dollars.

PROBLEM 4

Backpacking

Input file: `backin.txt`
Output file: `backout.txt`

Time and memory limits: 1 second, 1 GB

Norman is planning a backtracking trip through N towns, numbered 1 to N in the order that he will visit them. After careful planning, he knows that it will take D_i days to travel from town i to town $i+1$. Norman will start in town 1 and backpack all the way to town N, travelling for a total of $D_1 + D_2 + \ldots + D_{N-1}$ days.

Norman's backpack can fit up to K cans of food, and each day he will eat exactly one can. The price of food varies in each town; in town i, food costs C_i dollars per can. He can buy as many cans as he wants from each town as long as the total number of uneaten cans doesn't exceed K.

Norman will start in town 1 with an empty backpack, and wants to spend as little as possible on food by strategically choosing how much he buys in each town. What is the minimum total amount that he must spend to reach town N?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $2 \leq N \leq 200\,000$.
- $1 \leq K \leq 1\,000\,000$.
- $1 \leq D_i \leq K$ for all i. That is, Norman's backpack can store enough food for him to travel between consecutive towns on his trip.
- $D_1 + D_2 + \ldots + D_{N-1} \leq 1\,000\,000$. That is, Norman's trip is at most $1\,000\,000$ days in total.
- $1 \leq C_i \leq 20$ for all i.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (20 marks), $C_i \leq C_{i+1}$ for all i. That is, the towns are ordered from cheapest to most expensive.
- For Subtask 2 (30 marks), $K = 1\,000\,000$.
- For Subtask 3 (30 marks), $N \leq 1000$.
- For Subtask 4 (20 marks), no special rules apply.

Input

Your program must read input from the file `backin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `backin.txt` follows a specific format:

- The 1st line of input contains the integers N and K.
- The 2nd line of input contains $N-1$ integers describing the distances between towns. The ith of these is D_i.
- The 3rd line of input contains N integers describing the cost of food in each town. The ith of these is C_i.

Output

Your program must write a single integer to the file `backout.txt`: the minimum total amount (in dollars) that Norman must spend to reach town N.

Sample input 1

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

Sample input 2

```
5 1000000
2 2 2 2
5 3 4 1 2
```

Sample input 3

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

Sample output 1

```
16
```

Sample output 2

```
24
```

Sample output 3

```
25
```

Explanation

- In the 1st sample case, Norman's best strategy is:
  - fill his backpack by buying 5 cans at town 1 for $5 \times 2 = 10$ dollars
  - backpack from town 1 to town 2, which takes 4 days; when he arrives at town 2 he has 1 can left in his backpack
  - buy 2 cans at town 2 for $2 \times 3 = 6$ dollars; he now has 3 cans
  - backpack from town 2 to town 3, which takes 3 days; he arrives at town 3 with no cans left.

  This costs a total of $10 + 6 = 16$ dollars.
- In the 2nd sample case, Norman's best strategy is:
  - buy 2 cans at town 1 for $2 \times 5 = 10$ dollars
  - backpack from town 1 to town 2, arriving with no cans left in his backpack
  - buy 4 cans at town 2 for $4 \times 3 = 12$ dollars
  - backpack from town 2 to town 3, arriving with 2 cans left in his backpack
  - don't buy any cans at town 3
  - backpack from town 3 to town 4, arriving with no cans left in his backpack
  - buy 2 cans at town 4 for $2 \times 1 = 2$ dollars
  - backpack from town 4 to town 5, arriving with no cans left in his backpack.

  This costs a total of $10 + 12 + 2 = 24$ dollars.
- In the 3rd sample case, Norman's best strategy is:
  - buy 2 cans at town 1 for $2 \times 5 = 10$ dollars
  - backpack from town 1 to town 2, arriving with no cans left in his backpack
  - buy 3 cans at town 2 for $3 \times 3 = 9$ dollars; his backpack is now full
  - backpack from town 2 to town 3, arriving with 1 can left in his backpack
  - buy 1 can at town 3 for $1 \times 4 = 4$ dollars; he now has 2 cans
  - backpack from town 3 to town 4, arriving with no cans left in his backpack
  - buy 2 cans at town 4 for $2 \times 1 = 2$ dollars
  - backpack from town 4 to town 5, arriving with no cans left in his backpack.

  This costs a total of $10 + 9 + 4 + 2 = 25$ dollars.

PROBLEM 5

Tennis Robot II

Input file: `tennisin.txt`
Output file: `tennisout.txt`

Time and memory limits: 1 second, 1 GB

The cleaning robot at your local sports centre has been hard at work since they bought it several years ago. However, years of reprogramming has left its code a complete mess!

Currently, the sports centre has N bins used to store tennis balls. The ith bin starts with X_i tennis balls but has enough space to store as many extra balls as necessary.

The cleaning robot is currently programmed with M instructions. The jth instruction is to take one tennis ball from bin A_j and move it to bin B_j. The robot will carry out the instructions in order, and once it completes the final instruction it will loop back to the 1st one and continue. It will only stop if it is instructed to take a ball from an empty bin, at which point it will crash and shut down.

You want to know how long it will take for the robot to stop, so that you can fix it to do something more useful (it's not even cleaning, it's just moving balls between bins!).

How many instructions will the robot successfully complete before stopping, or will it run forever?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $2 \leq N \leq 200\,000$.
- $1 \leq M \leq 200\,000$.
- $1 \leq X_i \leq 1\,000\,000$ for all i.
- $1 \leq A_j, B_j \leq N$ and $A_j \neq B_j$ for all j.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (25 marks), $N, M \leq 1000$ and $X_i \leq 1000$ for all i.
- For Subtask 2 (40 marks), $A_j \neq A_k$ for all $j \neq k$. In other words, all instructions take tennis balls from different bins[1].
- For Subtask 3 (35 marks), no special rules apply.

Input

Your program must read input from the file `tennisin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `tennisin.txt` follows a specific format:

- The 1st line of input contains the integers N and M.
- The 2nd line of input contains N integers describing the number of balls that start in each bin. The ith of these is X_i.
- The next M lines describe the robot's instructions. The jth of these lines contains the two integers A_j and B_j.

[1] Sample input 1 satisfies the rules of subtask 2, but sample inputs 2 and 3 do not.

Output

Your program must write to the file `tennisout.txt`. If the robot will run forever your program must output `FOREVER`. Otherwise, it must output the number of instructions the robot will successfully complete before stopping.

Note: For students using C, C++ or Java, please note that the answer may exceed the maximum value that can be stored in an `int` *integer type. As such, you should consider using the* `long long` *integer type in C or C++, or the* `long` *integer type in Java. Please refer to the solution templates (which you can find on the* Templates & Downloads *page of the competition website) for more details. Python users do not need to consider this.*

Sample input 1

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

Sample input 2

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

Sample input 3

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

Sample output 1

```
4
```

Sample output 2

```
FOREVER
```

Sample output 3

```
9
```

Explanation

- In the 1st sample case, there are 4 bins and they start with 2, 1, 1, and 3 tennis balls. Here are the instructions that the robot executes and the number of tennis balls in the bins after each instruction.

Instruction	Successful?	Bin 1	Bin 2	Bin 3	Bin 4
Bin 1 to bin 2	Yes	1	2	1	3
Bin 3 to bin 1	Yes	2	2	0	3
Bin 2 to bin 4	Yes	2	1	0	4
Bin 1 to bin 2	Yes	1	2	0	4
Bin 3 to bin 1	No	–	–	–	–

  The robot successfully executes 4 instructions, then crashes on the next instruction when it tries to take a ball from bin 3, which is empty.

- In the 2nd sample case, the robot will continue executing instructions forever.
- In the 3rd sample case, the robot will successfully execute 9 instructions before crashing.

PROBLEM 6

Atlantis III: Twin Rivers

Input file: `twinin.txt`
Output file: `twinout.txt`

Time and memory limits: 1.5 seconds, 1 GB

After years of regulatory delays, King Triton has finally established the city of New Atlantis. To honour the old city, the citizens have chosen a location that features not one but two rivers running through the city.

The city can be modelled as 3 horizontal segments representing the strips of land that the residents live in, labelled Strip 1, 2 and 3. The strips of land are separated by rivers that are each 1 km wide, labelled River 1 and 2. The city is L km long from left to right and King Triton has built N bridges, each spanning one of the rivers. The ith bridge spans river R_i at a point B_i km from the left of the city. To prevent issues in the waterways, residents of New Atlantis are not permitted to swim and can only cross at bridges.

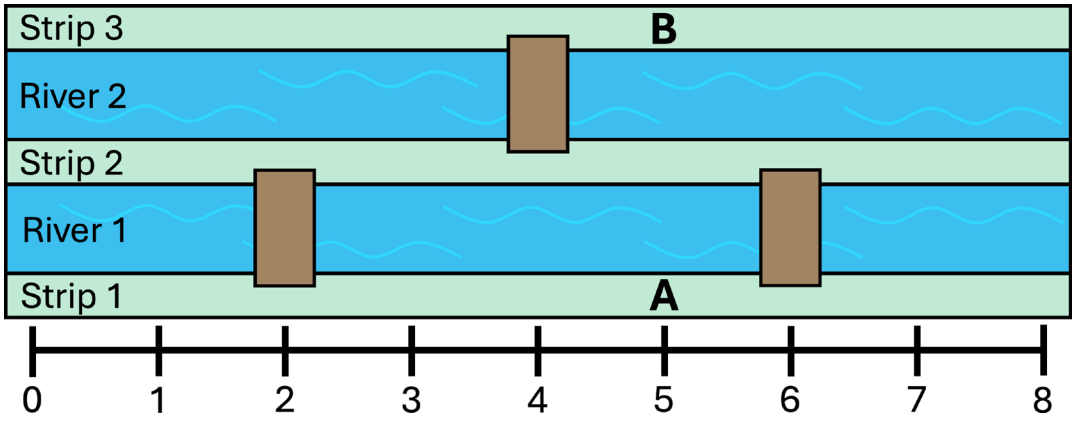

Figure 1: A possible city layout with 3 bridges. This layout corresponds to Sample Input 1.

Residents regularly travel between locations and always take the shortest route, travelling only on land or across bridges. King Triton has identified T trips that residents regularly take. The jth trip starts on Strip 1, at a point X_j km from the left of the city. The trip finishes on Strip S_j (either 2 or 3), also at a point X_j km from the left of the city.

For example in Figure 1, a trip from location A to B corresponds to a trip with $X_j = 5$ and $S_j = 3$. The shortest route for this trip is to walk 1 km right, then cross the 1 km-long bridge across River 1, then walk 2 km left, then cross the 1 km-long bridge across River 2, and finally walk 1 km right. This is a total of 6 km. The strips of land are very thin, and so the vertical distance walked within the strips is ignored. We only count the horizontal distance walked along the strips and the vertical distance walked when crossing bridges.

King Triton wishes to build one new bridge. The bridge must span only one of the rivers and must be built an integer number of kilometres from the left of the city, just like the existing bridges. After building the new bridge, it must be possible to complete every trip. Additionally, to save time for his citizens, he wishes to pick a location so that the sum of the distances of all trips will be as small as possible.

As King Triton's top advisor, you have been tasked with figuring out what this value is. What is the smallest possible sum of trip distances you can achieve after building one more bridge?

Subtasks and constraints

Your program will be graded using many secret tests. Every test follows some rules:

- $1 \leq N \leq 200\,000$.
- $1 \leq L \leq 1\,000\,000$.
- $1 \leq T \leq 200\,000$.
- $0 \leq B_i \leq L$ for all i.
- $1 \leq R_i \leq 2$ for all i.
- $0 \leq X_j \leq L$ for all j.
- $2 \leq S_j \leq 3$ for all j.
- $B_i \leq B_{i+1}$ for all i. That is, the bridges are given in increasing order of B_i.
- $X_j \leq X_{j+1}$ for all j. That is, the trips are given in increasing order of X_j.
- There is at least one bridge crossing the 1st river.
- No two bridges are the same. That is, $B_i \neq B_k$ or $R_i \neq R_k$ for all $i \neq k$.

The secret tests are divided into subtasks. Your program must correctly solve **every test** within a subtask to earn the marks for that subtask:

- For Subtask 1 (15 marks), $N, T, L \leq 2000$, $R_i = 1$ for all i and $S_j = 2$ for all j. That is, all existing bridges cross the 1st river and every trip finishes on Strip 2.
- For Subtask 2 (35 marks), $R_i = 1$ for all i and $S_j = 2$ for all j. That is, all existing bridges cross the 1st river and every trip finishes on Strip 2.
- For Subtask 3 (30 marks), $R_i = 1$ for all i and $S_j = 3$ for all j. That is, all existing bridges cross the 1st river and every trip finishes on Strip 3.
- For Subtask 4 (20 marks), no special rules apply.

Input

Your program must read input from the file `twinin.txt`. When testing on your own computer, this file must be placed in the same folder as your program. We strongly recommend using the solution templates (which you can find on the *Templates & Downloads* page of the competition website) to help you with input and output.

The file `twinin.txt` follows a specific format:

- The 1st line of input contains the integers N and L.
- The next N lines describe the locations of the existing bridges. The ith of these lines contains the two integers B_i and R_i.
- The next line contains the integer T.
- The next T lines describe the trips. The jth of these contains the two integers X_j and S_j.

Output

Your program must write a single integer to the file `twinout.txt`: the smallest possible sum of trip distances (in km) after building one additional bridge.

Note: For students using C, C++ or Java, please note that the answer may exceed the maximum value that can be stored in an `int` *integer type. As such, you should consider using the* `long long` *integer type in C or C++, or the* `long` *integer type in Java. Please refer to the solution templates (which you can find on the* Templates & Downloads *page of the competition website) for more details. Python users do not need to consider this.*

Sample input 1

```
3 8
2 1
4 2
6 1
4
3 3
4 3
5 3
5 2
```

Sample input 2

```
2 6
1 1
5 1
4
0 2
3 2
3 2
6 2
```

Sample input 3

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

Sample output 1

```
13
```

Sample output 2

```
8
```

Sample output 3

```
10
```

Explanation

- The 1st sample case is shown in Figure 1. The smallest sum of trip distances can be achieved by building a bridge across River 1 at the point 4 km from the left end of the city:
  - The 1st trip $(X_1 = 3, S_1 = 3)$ reduces from 6 km to 4 km.
  - The 2nd trip $(X_2 = 4, S_2 = 3)$ reduces from 6 km to 2 km.
  - The 3rd trip $(X_3 = 5, S_3 = 3)$ reduces from 6 km to 4 km.
  - The 4th trip $(X_4 = 5, S_4 = 2)$ remains 3 km long.
- In the 2nd sample case, every trip finishes on Strip 2. Initially, the trip distances are 3 km, 5 km, 5 km, and 3 km respectively for a total of 16 km. Building a bridge across River 1 at the point 3 km from the left of the city would reduce the sum of distances to 8 km, which is the smallest possible.
- In the 3rd sample case, it is not originally possible to complete the trips because there is no bridge crossing River 2. One option is to build a bridge crossing River 2 at the point 3km from the left of the city. The trip distances would be 6 km and 4 km, giving a sum of 10 km which is the smallest possible.