

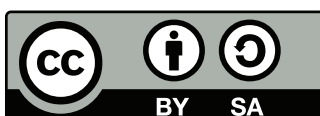
# BAPC 2023 Preliminaries

*Preliminaries for the  
2023 Benelux Algorithm Programming Contest*



## Problems

- A Anti-Tetris
- B Better Dice
- C Cheap Flying
- D Determining Duos
- E Exceeding Limits
- F Finding Forks
- G Gathering Search Results
- H Hacky Ordering
- I Idle Terminal
- J Just a Joystick
- K King's Keep
- L Losing Leaves
- M Monorail



Copyright © 2023 by The BAPC 2023 jury. This work is licensed under the Creative Commons Attribution-ShareAlike 4.0 International License.  
<https://creativecommons.org/licenses/by-sa/4.0/>

## A Anti-Tetris

In the game *Tetris*, the goal is to position blocks falling down a grid as well as possible. *Before* the block falls down, the player can shift the block to the left and right, and rotate it in steps of 90 degrees. Then, the block falls down vertically until it hits another block. Completely filling a row removes this row from the grid, clearing up space for more falling blocks.

You have played this game one too many times, and to shake things up, you decide to play *Anti-Tetris*: instead of controlling the positioning of the blocks falling down, the goal is to design a Tetris grid that will perfectly fit a given block. That is, a grid such that after optimally positioning the new block, all rows of the grid are cleared and no filled cells remain.

As an example, consider the first sample case, shown in Figure A.1. The input block can be rotated clockwise 90 degrees and shifted left to make it fit exactly and clear all rows of the grid once it touches down.

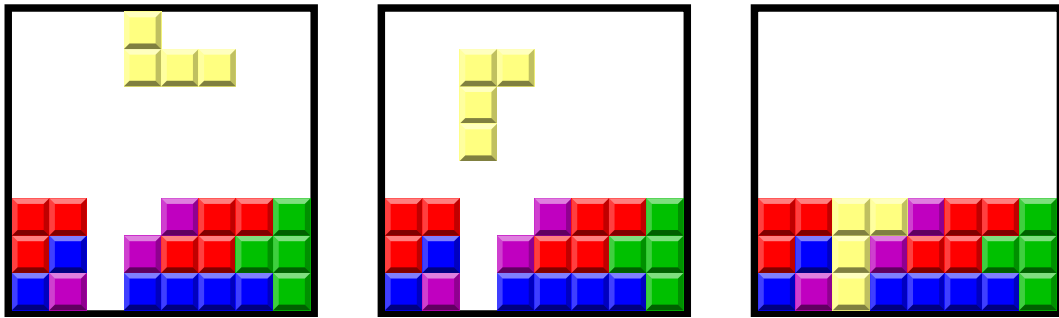


Figure A.1: Visualization of the first sample case. The falling block (the input, light yellow) perfectly fits in the Tetris grid (the output, other colours).

### Input

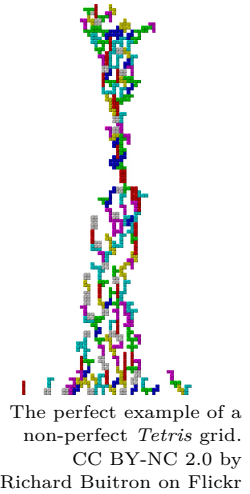
The input consists of:

- One line with two integers  $h$  and  $w$  ( $1 \leq h, w \leq 100$ ), the height and width of the Tetris block that is about to enter the grid.
- $h$  lines with  $w$  characters, each character being either '#' or '.', representing a filled or unfilled cell of the block, respectively.

The input block is a single orthogonally<sup>1</sup> connected component and exactly fits in the  $w \times h$  bounding box, i.e. the first and last row and column contain at least one '#'.

<sup>1</sup>Two cells are orthogonal neighbours if and only if they are horizontal or vertical neighbours.

Time limit: 1s



## Output

If there exists no Tetris grid that perfectly fits the input block, output “impossible”. Else, output a grid such that placing the the input block optimally removes all rows, in the following format:

- Two integers  $h, w$  ( $1 \leq h, w \leq 1000$ ), the height and width of the Tetris grid.
- $h$  strings with  $w$  characters, each character being either ‘#’ or ‘.’, representing a filled or unfilled cell in the Tetris grid, respectively.

A row in the output grid may not be completely filled before the block is added, since such a row would already have been removed by the game.

Note that it is not required to print empty rows at the top of the output grid, since the block can be rotated and shifted to the left and right *before* it falls down.

If there are multiple valid solutions, you may output any one of them.

### Sample Input 1

```
2 3
#..
###
```

### Sample Output 1

```
3 8
##..####
##.#####
##.#####
```

### Sample Input 2

```
2 3
.##
##.
```

### Sample Output 2

```
impossible
```

### Sample Input 3

```
3 3
#..
##.
###
```

### Sample Output 3

```
5 7
.....
.....
##...##
###.##
####.##
```

## B Better Dice

Time limit: 1s

The latest Table-Top Role Playing Game is out now: *Better Dice*. Unlike all other TTRPGs, this one is all about dice. In fact, it is all about the *better die*: decisions are made, friendships gained and lost, fights fought, battles won, all based on who has the *better die*.

This game uses special  $n$ -sided dice where each of the  $n$  faces has the same probability of being rolled. Moreover, each die has its own special set of  $n$  numbers on the faces.

While playing *Better Dice* you ended up in a very precarious situation where you must absolutely have a *better die* than your opponent, that is, you must roll higher than your opponent. Given both your die and your opponent's die, decide who is more likely to roll a higher number.



A twenty-sided die with a special set of numbers on the faces. CC BY 4.0 by hamstermann on Thingiverse

### Input

The input consists of:

- One line with an integer  $n$  ( $1 \leq n \leq 1000$ ), the number of sides on each die.
- Two lines, each with  $n$  integers  $d$  ( $1 \leq d \leq 10^9$ ), the values on one of the dice.

### Output

Output “first” if the first die is more likely to roll a higher number than the second die.

Output “second” if the second die is more likely to roll a higher number than the first die.

Output “tie” if they are both equally likely to come up higher than the other.

#### Sample Input 1

2	tie
4 6	
5 5	

#### Sample Output 1

#### Sample Input 2

6	second
1 2 3 4 5 6	
7 6 5 4 3 2	

#### Sample Output 2

#### Sample Input 3

3	first
2 2 2	
1 1 8	

#### Sample Output 3

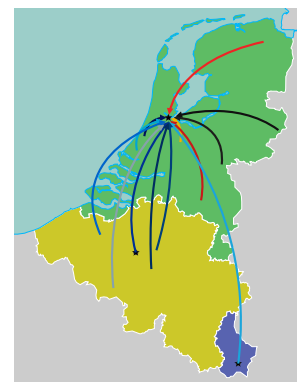
This page is intentionally left blank.

## C Cheap Flying

Time limit: 3s

You are part of the organization of BAPC, and you are in charge of flight operations. From time to time, the jury members of BAPC need to fly from the remote headquarters of BAPC to the current location of the operation that BAPC performs, which is currently Amsterdam. Luckily there is an airline that serves the route from BAPC's headquarters to Amsterdam. The BAPC organization has a contract with that airline that ensures that you pay the same fixed amount for each flight. If your judges need to fly the route often, these costs can add up and become really high. To avoid this, you figured out that you could also simply buy your own aircraft. Once you own your own aircraft, you can either fly with this new shiny equipment which costs some fixed price per flight for fuel and the like, or alternatively you could still fly with the airline.

The problem now is that you have no idea what the judges are doing! They are so incredibly unpredictable and always only do random and very complicated things. So random, that you cannot decide beforehand what you should do. So you need to make your decision on the fly, even though this may prevent you from making the cost-optimal decision up-front. Still, you do not want to be too loose with your spending: you set yourself the constraint that you spend at most twice as much as you would have if you exactly knew how many flights the judges would make in advance.



The unpredictability of the jury members would be even worse, if they would all fly from their own universities, instead of from the headquarters of BAPC.  
CC BY-SA 4.0 by Shaund on Wikimedia Commons, modified

### Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads from the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends one line with three integers  $a$ ,  $b$ , and  $c$  ( $0 \leq a, b, c \leq 10^6$ ), the cost of one flight with the airline, the one-time cost of buying an aircraft, and the cost for each flight using your own aircraft.

Then, the interactor sends either the input “flight” or the input “end”. For every input “flight”, you need to provide either “airline” if you want fly with the airline for cost  $a$ , “buy” to buy an aircraft and use it for a single flight for cost  $b + c$ , or “self” if you want to use your own aircraft for cost  $c$ . You can only use the last option if you have bought an aircraft before, and you can only buy an aircraft once.

The interaction ends when you receive the input “end”, which is after at most  $10^4$  rounds.

The interactor is adaptive, and may send fewer or more flights based on your output.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Read	Sample Interaction 1	Write
5 50 2 flight		
	airline	
flight		
	airline	
flight		
	airline	
end		

Read	Sample Interaction 2	Write
5 8 1 flight		
	buy	
flight		
	self	
end		



## D Determining Duos

Time limit: 3s

As a coach of  $2n$  students, you are making  $n$  duos (teams of two) for the upcoming programming contest season. After the duos have been created, they will participate in  $r$  contests, each about a different topic: DP, graphs, geometry, etc. You already ran a set of internal selection contests to rank the students, and from this you were able to rank all the students with a unique integer score between 1 and  $2n$  inclusive for each topic, with  $2n$  being the best.

When a duo participates in a contest on a given topic, their score will be the maximum of the two scores of the two students for this topic.

You think it would be amazing if summed up over all duos and contests, your students could achieve a total score of at least  $\frac{1}{2}rn(3n + 1)$ . Is this possible?

### Input

The input consists of:

- One line with two integers  $n$  and  $r$  ( $1 \leq n \leq 4000$ ,  $1 \leq r \leq 100$ ), the number of duos and the number of topics.
- $r$  lines, the  $i$ th of which contains  $2n$  integers  $x_{i,1}, \dots, x_{i,2n}$  ( $1 \leq x_{i,j} \leq 2n$  for each  $i, j$ ) where  $x_{i,j}$  is the score of student  $j$  on topic  $i$ .

### Output

If it is possible to make duos such that the total score over all duos and contests is at least  $\frac{1}{2}rn(3n + 1)$ , output “possible”. Otherwise, output “impossible”.

Sample Input 1	Sample Output 1
2 2 1 2 3 4 1 2 3 4	possible

Sample Input 2	Sample Output 2
2 2 1 2 3 4 4 1 2 3	possible

BAPC 2023 Preliminaries  
Preliminaries for the  
2023 Benelux Algorithm Programming Contest



- Problems
- A Anti-Tetris
  - B Better Dice
  - C Cheap Flying
  - D Determining Duos
  - E Exceeding Limits
  - F Finding Forks
  - G Gathering Search Results
  - H Hacky Ordering
  - I Idle Terminal
  - J Just a Joystick
  - K King's Keep
  - L Losing Leaves
  - M Monocall

Typical example of a programming contest that you are coaching your students for.

**Sample Input 3**

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

**Sample Output 3**

```
impossible
```

E Exceeding Limits

Time limit: 8s

Tim needs to reach the Binary Analog Probing Conference (BAPC) on time, but he is running late. He is not sure if he can even make it on time without exceeding the speed limit! He does not like speeding, so he would like to minimize the amount that he needs to speed and plans his route accordingly. If he decides to speed by  $x$  km/h, he will exceed the speed limit everywhere by exactly  $x$  km/h.

Help Tim find the minimal amount that he needs to speed by to get to the BAPC in time.

As an example, consider the first sample case. Without speeding, Tim will take  $\frac{400}{40} + \frac{300}{20} = 25$  hours to drive from intersection 1, via intersection 3, to intersection 4. In order to arrive in time, he will need to exceed the speed limit by 10 km/h, in which case his driving time will be  $\frac{400}{40+10} + \frac{300}{20+10} = 18$  hours, following the same route.



Tim’s arch-nemesis:  
the *trajectcontrole*.  
CC BY-NC-SA 2.0 by  
DutchRoadMovies on Flickr

Input

The input consists of:

- One line with three integers  $n$ ,  $m$ , and  $t$  ( $2 \leq n \leq 10^4$ ,  $1 \leq m \leq 10^5$ ,  $1 \leq t \leq 10^5$ ), the number of intersections, the number of roads, and the time within which Tim needs to reach his destination.
- $m$  lines, each with four integers  $a$ ,  $b$ ,  $\ell$ , and  $v$  ( $1 \leq a, b \leq n$ ,  $a \neq b$ ,  $1 \leq \ell, v \leq 10^5$ ). Each line indicates a bidirectional road between intersections  $a$  and  $b$  with length  $\ell$  in km and speed limit  $v$  in km/h.

The intersections are numbered between 1 and  $n$ , inclusive.

Tim will start at intersection 1 and drive to intersection  $n$ , which is guaranteed to be reachable.

Output

Output how much Tim needs to exceed the speed limit, in km/h. If Tim can reach his destination without speeding, output 0.

Your answer should have an absolute or relative error of at most  $10^{-6}$ .

Sample Input 1	Sample Output 1
4 4 18 1 2 800 40 1 3 400 40 4 2 500 50 4 3 300 20	10

**Sample Input 2**

```
4 3 100
1 2 300 15
2 3 500 20
3 4 300 30
```

**Sample Output 2**

```
0
```

**Sample Input 3**

```
4 4 10
1 2 200 50
2 3 300 30
2 3 400 15
3 4 500 50
```

**Sample Output 3**

```
56.9041576
```

F Finding Forks

Time limit: 1s

Your cutlery drawer contains many types of forks. Each with their own purpose, and each with its own place in the cutlery drawer. After a nice dining party with all your friends, disaster struck! You put all the used forks in the dishwasher, but now you are unsure where to put back some of the forks, because at least two places in the cutlery drawer are empty! And worse, you do not remember which type of fork belongs where!

What is the minimum number of forks that must have been in the dishwasher to cause this confusion?



A small selection of the many types of forks that you own. CC BY-SA 3.0 by Mark Taff on Wikimedia Commons

Input

The input consists of:

- One line with an integer  $n$  ( $2 \leq n \leq 10^5$ ), the number of types of forks.
- One line with  $n$  integers  $a$  ( $1 \leq a \leq 10^9$ ), the number of forks of each type.

Output

Output the minimum number of forks that must have been in the dishwasher.

Sample Input 1	Sample Output 1
3 4 9 5	9

Sample Input 2
10 18 39 5 12 1000000000 54 23 11 123 31415

Sample Output 2
16

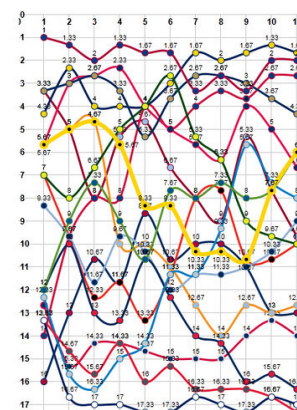
This page is intentionally left blank.

# G Gathering Search Results

Time limit: 3s

You are making a new search engine for algorithms, called the Benelux Algorithm Preview Collector. All the marketing has been done, and you have plenty of investors, but there is one small problem: you have not written any code yet! As there are only five hours left until the product launch, you decide that there is not enough time to implement your own ranking algorithm. Instead, whenever a user searches for an algorithm, you just forward it to the  $k$  most popular other search engines and use their results.

Each of these  $k$  search engines gives a list of  $n$  results, ordered by estimated relevance. Surprisingly, it turns out that for every search term you try, the different search engines always give the same  $n$  results, only the order differs between the search engines! You just need to find a way to combine the  $k$  different rankings into a single ranking.



Throwback to when you still ranked search results manually.  
CC BY-SA 2.0 by Matthew Bernhardt on Flickr

To do this, you decide to give each possible combined ranking a cost: for each result  $r$  and search engine  $s$ , if your combined ranking puts  $r$  at the position  $i$ , and search engine  $s$  puts  $r$  at position  $j$ , then you incur a cost of  $(i - j)^2$ . The total cost of a combined ranking is the sum of all  $k \cdot n$  costs computed this way.

As an example, consider the second sample case. The first search engine returns the same result as the sample output, which has cost 0. The second search engine swaps two adjacent results, which has cost  $1 + 1 = 2$ . For the third search engine, the first result is moved two positions back, and the other two are shifted one position to the front, which has cost  $4 + 1 + 1 = 6$ . The total cost is then  $0 + 2 + 6 = 8$ , which is the minimal possible cost over all possible orders.

What is the order in which BAPC should output the results, such that the total cost is minimized?

## Input

The input consists of:

- One line containing two integers  $n$  and  $k$  ( $1 \leq n \leq 5000$ ,  $1 \leq k \leq 100$ ), the number of results, and the number of search engines.
- $k$  lines, each containing  $n$  integers, denoting the results as ordered by one of the search engines. It is guaranteed that each line contains every integer from 1 to  $n$  exactly once.

## Output

Output the combined order of the results, such that the cost is minimized.

If there are multiple valid solutions, you may output any one of them.

**Sample Input 1**

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

**Sample Output 1**

```
2 1
```

**Sample Input 2**

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

**Sample Output 2**

```
1 2 3
```

**Sample Input 3**

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

**Sample Output 3**

```
1 2 4 5 3
```



## H Hacky Ordering

Time limit: 2s

You have been asked to sort! Again! For the bazillionth time! Not even numbers, but strings! Ugh! Do people still not have this in their standard library? Why do you even need to learn this? Who even uses a language without `sort` function?

Clearly, you have not been paying attention in class for such a stupid ubiquitous function, but now you have been asked to implement it! Without calling `sort`! But you just cannot!

But wait! You have a better approach: what if you just assume that the list is sorted already? The order of the characters in the alphabet is arbitrary anyway... So, instead of sorting the list, you want to determine whether there exists some order of the characters of the alphabet such that the list of strings is sorted according to this order.

Note that when a string is a prefix of some longer string, the shorter string should be sorted before the longer string.

```
#include <algorithm>
import java.util.Arrays;

const sort = (names) =>
  sorted(
    Arrays.sort(
      sort(names.begin(),
        names.end())
    ).sort()
  );

If only you were allowed
to use this universal
sorting function...
```

### Input

The input consists of:

- One line with an integer  $n$  ( $1 \leq n \leq 10^5$ ), the number of strings.
- $n$  lines, each with a string.

The strings only consist of English lowercase letters (a–z).

The total number of characters in the  $n$  strings is at most  $10^5$ .

The strings are not necessarily distinct.

### Output

If it is impossible to determine an order of the alphabet, output “impossible”.

If it is possible, output a permutation of the 26 letters of the English alphabet according to which the strings are sorted.

If there are multiple valid solutions, you may output any one of them.

#### Sample Input 1

```
7
c
plusplus
csharp
python
php
java
javascript
```

#### Sample Output 1

```
cpsyhjabdefgiklmnoqrtuvwxz
```

**Sample Input 2**

```
4
aa
ba
ab
bb
```

**Sample Output 2**

```
impossible
```

**Sample Input 3**

```
5
YYY
YYYY
z
xx
xx
```

**Sample Output 3**

```
qwertyuiopasdfghjklzxcvbnm
```

**Sample Input 4**

```
2
aa
a
```

**Sample Output 4**

```
impossible
```

I Idle Terminal

Time limit: 1s

It is migration day at the Big Administration Processing Company: the database containing all administrative documents of all clients needs to be migrated to the latest version of the database software. Quite some things have changed over the past years, and this software has not been upgraded, so the number of migration jobs is high.



The IT Team praying that the migrations will complete successfully.  
From r/pcmasterrace

The migration jobs run in parallel on the multicore server machine in a first-come-first-serve fashion: every time a core is done running a job, it starts running the first job that has not yet started. The Idle Terminal (IT) Team is sitting huddled around the terminal window, eagerly awaiting the dopamine boost when another migration job completes successfully and a message is printed to the terminal.

The IT Team starts breaking out in sweat when nothing changes on the terminal for quite a long time. Did the connection hang? Has the server soft-locked? Did all jobs get stuck in infinite loops?? On the other hand, some migration jobs really do have a long duration, and it may simply be a coincidence that there are only long-running jobs active at the time. To calm down the IT Team, you decide to compute the longest time that goes by without seeing a new message on the terminal, starting from the moment that the first migration jobs start running.

Input

The input consists of:

- One line with two integers  $n$  and  $k$  ( $1 \leq n, k \leq 10^5$ ), the number of migration jobs and the number of cores in the server machine.
- One line with  $n$  integers  $d$  ( $1 \leq d \leq 100$ ), the duration of each job in the order that they are processed.

Output

Output the longest time that goes by without seeing a new message on the terminal.

Sample Input 1	Sample Output 1
2 2 3 7	4
Sample Input 2	Sample Output 2
5 10 1 2 3 4 5	1

**Sample Input 3**

4 1
2 10 6 4

**Sample Output 3**

10
----

**Sample Input 4**

6 2
3 5 8 10 4 1

**Sample Output 4**

6
---

**Sample Input 5**

6 3
2 4 6 6 6 6

**Sample Output 5**

2
---

J Just a Joystick

Time limit: 1s

You just got the high score when playing *Battlezone Asteroids Pac-Centipede* on an arcade machine! On the “Game Over” screen, you can enter your initials, one letter at a time. This seems to be a very modern arcade machine: whereas the original arcade machines only allowed entering three initials, this machine allows many more. However, to select the letters, you have access to just a joystick. For every letter, you need to move the joystick up or down to cycle between the letters (wrapping around between ‘Z’ and ‘A’, in both directions) and move it to the right to move to the next letter.

It appears that the initials of the previous high-score winner are still filled in. Entering your own initials is going to take some time, and you want to know exactly how long. How many times do you need to you move the joystick up or down to enter your own initials, if you do so in the most efficient way?



A girl playing *Battlezone Asteroids Pac-Centipede*. CC BY-SA 2.0 by Sergiy Galyonkin on Flickr

Input

The input consists of:

- One line with an integer  $n$  ( $1 \leq n \leq 10^5$ ), the number of letters available to enter your initials.
- One line with a string of length  $n$ , the initials of the previous high-score winner.
- One line with a string of length  $n$ , the initials that you want to enter.

The strings only consist of English uppercase letters (A–Z).

Output

Output the minimum number of times you should move the joystick up or down to enter your own initials. This does *not* include the number of times that you need to move the joystick to the right.

Sample Input 1	Sample Output 1
3 RGK MPS	22

Sample Input 2	Sample Output 2
5 ABIYZ YZIAB	8

This page is intentionally left blank.

K King’s Keep

Time limit: 1s

King Carl’s kingdom contains  $k$  keeps (commonly called castles). Coordinates of keeps are known, and King Carl considers himself convinced that it could be convenient to choose a central keep as King Carl’s residence.

Critically, King Carl considers that the average cost to carry commands from King Carl’s residence to King Carl’s other keeps should be small.



King Carl’s keep of choice: created with contours of a capital ‘K’.  
CC BY-SA 2.0 by David Dixon on  
geograph.co.uk, modified

Compute the minimal average Euclidean distance<sup>2</sup> from his residence keep to the other keeps if King Carl chooses his residence optimally.

Input

The input consists of:

- One line with an integer  $k$  ( $2 \leq k \leq 1000$ ), the number of keeps.
- $k$  lines, each with two integers  $x$  and  $y$  ( $|x|, |y| \leq 1000$ ), the coordinates of the keeps.

It is guaranteed that all keeps are at distinct locations.

Output

Output the minimal possible average distance from the keep that is chosen as the king’s residence to all other keeps.

Your answer should have an absolute or relative error of at most  $10^{-6}$ .

Sample Input 1	Sample Output 1
3 0 0 9 9 0 9	9

Sample Input 2	Sample Output 2
5 -3 5 6 8 1 2 5 -4 -7 -9	8.405705684

<sup>2</sup>The Euclidean distance between two points is the length of a straight line segment between these points.

This page is intentionally left blank.



## L Losing Leaves

Time limit: 8s

Here at the Benelux Advanced Phone Company (BAPC), we are the proud owners of the largest telephone network in the Benelux area. Unfortunately, our network has become too large for us to manage properly. As such, we have decided to sell part of our network.

The network of the BAPC consists of interconnected transmission nodes. One transmission node is marked as the central switch. All other nodes have exactly one upstream transmission node. For each transmission node, if we follow the upstream connections, we will finally end up at the central switch. A node is considered a customer transmission node when it is a leaf, i.e. when it has no downstream nodes.

When selling parts of our network, integrity must be maintained. That means that whenever we sell a transmission node  $X$ , we also have to sell nodes downstream of  $X$ .

Overall, BAPC decided to sell exactly  $k$  transmission nodes. While there may be many options to choose these  $k$  nodes, we actually want to make our lives as easy as possible: After selling, we want to minimize the number of customer transmission nodes in our network, while maintaining the network's integrity.

As an example, consider the second sample case, visualized in Figure L.1. The three striped red nodes are sold, and the two bold green nodes are the remaining customer nodes.

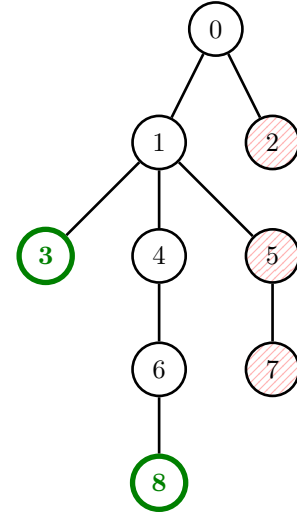


Figure L.1: Visualization of the second sample input.

### Input

The input consists of:

- One line with two integers  $n$  and  $k$  ( $0 \leq k < n \leq 10^6$ ), the number of transmission nodes, and the number of nodes to sell.
- $n - 1$  lines, the  $i$ th of which contains one integer  $p_i$  ( $0 \leq p_i < i$ ) indicating that transmission node  $i$  ( $1 \leq i < n$ ) has an outgoing connection to node  $p_i$ .

The transmission nodes are numbered from 0 to  $n - 1$ , inclusive. Node 0 is always the central switch.

### Output

Output the minimum number of customer transmission nodes after selling  $k$  transmission nodes. Note that if the central switch is the only remaining node, it also counts as a customer node.

**Sample Input 1**

```
5 2
0
0
1
1
```

**Sample Output 1**

```
1
```

**Sample Input 2**

```
9 3
0
0
1
1
1
1
4
5
6
```

**Sample Output 2**

```
2
```

## M Monorail

Time limit: 2s

You were just about to go on a nice long holiday to the south, but as always, the trains are delayed. This time, a cargo train derailed in the Gotthard Base Tunnel through the Alps, completely taking one of the two tubes out of service for several months. Luckily, after some initial repairs, the other tube is in service again for cargo traffic.



A train about to enter the tunnel.  
From pxfuel.com

Since it is now only a one-track connection, multiple trains in the same direction can closely follow each other, but trains in opposite directions can not pass each other. This also means that trains going north can only enter once all trains going south have exited the tunnel, and vice versa.

Today, there are  $n$  cargo trains that want to drive through the tunnel. Each train arrives at one of the ends at a given time, and takes exactly  $d$  minutes to drive through the tunnel at a constant speed.

Even though this is not your responsibility, you decide to make a schedule for today's trains. You will decide for each train how long it has to wait at its entrance portal before it can enter the tunnel. Your goal is to minimize the sum of waiting times at the entrance portals over all trains.

For simplicity, you assume that trains are short compared to the length of the tunnel and can be approximated by points travelling over a line segment.

### Input

The input consists of:

- One line with an integer  $n$  and  $d$  ( $1 \leq n \leq 500$ ,  $1 \leq d \leq 10^9$ ), the number of cargo trains and the duration that each train needs to drive through the tunnel, in minutes.
- $n$  lines, each containing a character  $s$  and an integer  $t$  ( $s \in \{'N', 'S'\}$ ,  $0 \leq t \leq 10^9$ ), indicating whether this train starts at the north or south portal, and the number of minutes after the start time at which this train arrives.

It is guaranteed that trains are sorted by arrival time. For trains with equal arrival time, the trains coming from the north are listed before the trains coming from the south.

### Output

Output the minimal sum of waiting times at the tunnel entrance portals over all trains, in minutes.

**Sample Input 1**

```
3 5
N 0
S 4
N 8
```

**Sample Output 1**

```
3
```

**Sample Input 2**

```
4 10
N 5
N 10
S 10
N 15
```

**Sample Output 2**

```
15
```

**Sample Input 3**

```
4 10
S 0
N 10
N 10
S 20
```

**Sample Output 3**

```
0
```

**Sample Input 4**

```
4 10
N 0
S 5
S 5
S 5
```

**Sample Output 4**

```
15
```

