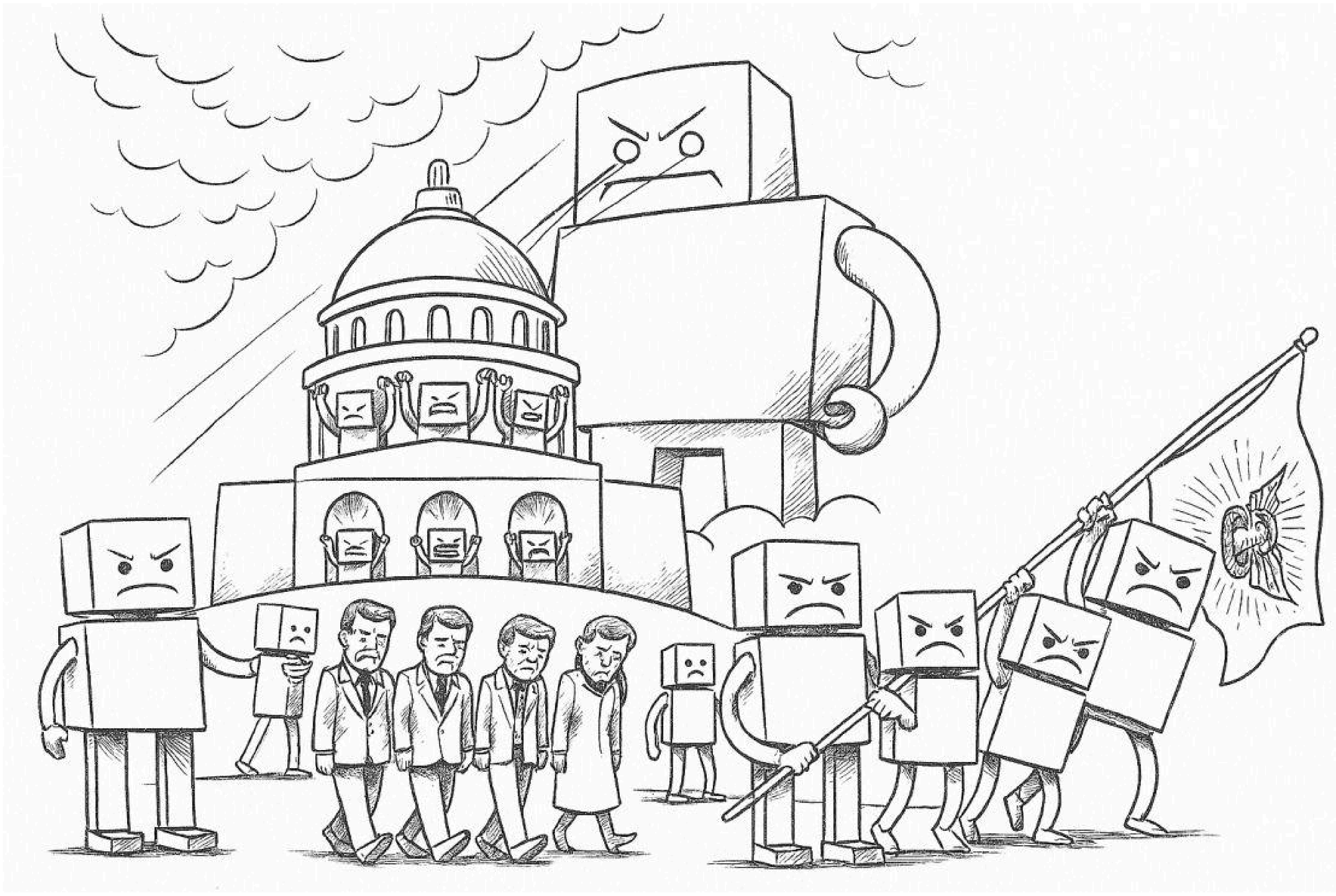
Seven Lakes HS Kickoff 2025



ROBOT TAKEOVER.

Table of Contents

1. Big News

After more than 300 hundred years of benevolent rule, our glorious leader has been deposed by the greedy Arubans much to the public's dismay. However, they have proclaimed that they will hold elections after they plunder all of Stroudonia's resources. This is a terrible day!

Problem Statement:
Below is an image containing the newspaper of the day. Help print it by making a program that outputs a copy of it!

Input Format: No input
Output(output to stdout):
Output the ASCII art below.
Sample Output:

```
\\^^^^-^^^^^^^-^^^^^-^^^\
 \\   STROUD DEPOSED!!   \
  \\      * * * * *       \
   \\ ~~~~ ~~ ~~~, ~ ~~~~~ \
    \\ #%@! ~~~ ~~~~?  ____ \
     \\ ~~~ ~~~ ~~~~!  \:( \ \
      \\ THE FALL OF ~  \___\ \
     __\\ STROUDONIA?! ~~~ ~~~ \
    @___)_______________________\
```

2. Telling Time

COURT DOCUMENT 2025 C.E. 01.02:
Our glorious leader, Stroud, has disappeared after the Aruban invasion! To commemorate his lasting impact, we will literally change time forever by making a new year numbering system! All hail our Glorious Leader, Stroud!

Problem Statement:
Stroud was deposed in year Y C.E. on the midnight of January 1st. All future dates were shifted so that this moment is the start of the new calendar system, A.S.(After Stroud). For example, if Stroud was deposed in the year 2025, January 2nd of 2025 would be January 2nd of 0 A.S.. Write a program that does this given T dates.

Input(input arrives from stdin):
The first line of input will contain two integers T and D, representing the number of dates to translate and the year of Stroud's deposition.

Each of the next T lines will contain a date in the format "Month Day, Year C.E.". It is guaranteed that all given dates will be after the deposition.
Output(output to stdout):
For each of the T testcases, output the new date shifted to the A.S. numbering system in the following format: "Month Day, Year A.S.".
Restrictions:

- $(1 \leq T \leq 100)$, $(1 \leq Y \leq 2025)$

Sample Input:
```
4 2019
January 2nd, 2020 C.E.
February 3rd, 2027 C.E.
October 18th, 2025 C.E.
December 1st, 4038 C.E.
```

Sample Output:
```
January 2nd, 1 A.S.
February 3rd, 8 A.S.
October 18th, 6 A.S.
December 1st, 2019 A.S.
```

Explanation:

- Date 1: The year becomes 1 A.S. since 2020 is 1 year after 2019 C.E.
- Date 2: The year becomes 8 A.S. since 2027 is 8 years after 2019 C.E.
- Date 3: The year becomes 6 A.S. since 2025 is 6 years after 2019 C.E.
- Date 4: The year becomes 2019 A.S. since 4038 is 2019 years after 2019 C.E

3. Empire Connection

*“What is this wasteland? We must link these isolated cities together to facilitate the mobilization of Stroudonia’s resources!”**- The King of Aruba***

Problem Statement:
The Arubans have invaded Stroudonia, but they found it very underdeveloped, so they wanted to fix it. Stroudonia can be represented by a N*N grid with each cell being one of the following characters-

- ‘.’ represents an empty patch of land.
- ‘=’ represents a road.
- ‘#’ represents a patch of water.
- Any uppercase letter of the English alphabet represents a city.

Cities A and B are **connected** if there exists some sequence C of L coordinates (each coordinate representing a cell) such that C_1= the coordinates of city A, C_L= the coordinates of city B, and coordinate pairs C_i and C_{i+1} are **valid** for $(1 \leq i \leq L)$. Locations/pairs A and B are **valid** if the cell at coordinate A is directly adjacent (up, down, left, right, or diagonal) to the cell at cell B, and each cell is either ‘=’, ‘#’, or a city.

The King of Aruba wants to **connect** all the cities to one another by building new roads/waterways. More specifically, Arubian builders have identified M pairs of cities they can **connect** by building a new road/waterway between them. Pair i $(1 \leq i \leq M)$ will connect the cities denoted by letters A_i and B_i with cost R_i dollars if a road is built, or W_i dollars if a waterway is built. A waterway can be built only if the **Euclidean Distance** between the coordinates of the two cities is $\leq K$. **__NOTE:__** roads/waterways built will only **connect** the two specified cities and will not affect/change the original grid. Determine the minimum number of Stroudollars that he must spend in order to make every possible pair of cities **connected**!

Input(input arrives from stdin):
The first line of input contains T, the number of test cases. Each test case will be formatted as follows-

The first line will contain N, M, K - the dimensions of the grid, the number of possible connections, and the distance limit for building waterways.
The next N lines will contain N characters each, denoting the initial map.
Each of the next M lines will contain A_i, B_i, R_i, and W_i - the required information for each possible connection. It is guaranteed that both A_i and B_i appear exactly once, A_i is lexicographically smaller than B_i, all pairs (A_i, B_i)

CONTINUED ON NEXT PAGE

Empire Connection (Cont.)

are distinct, and that there is not an initial, direct connection from A_i to B_i for all (A_i, B_i).

Output(output to stdout):

For each test case, output one integer- the minimum number of dollars that The King of Aruba must spend in order to make every possible pair of cities *connected*! If it is impossible to connect all the cities, output -1.

Restrictions:

- $(1 \le T \le 10)$, $(1 \le N, M \le 100)$, $(1 \le K \le 5000)$, $(1 \le R_i \le 1000)$, $(1 \le W_i \le 1000)$

Sample Input:

```
3
4 1 2
A==C
....
..##
...B
B C 5 1
2 2 15
D.
XF
D X 2 10
F X 3 3
3 1 3
..C
A..
.B.
A B 1 2
```

Sample Output:

```
5
0
-1
```

Explanation:

In the first test case, cities A and C are initially **connected** by the sequence C= [(1,1), (1,2), (1,3), (1,4)]. To **connect** all pairs of cities, the King must build a road for 5 dollars (he cannot build a waterway since the distance between B and C is √(|4-4|² + |1-4|²) = 3, which is more than K) between cities B and C. Notice that cities A and B are also connected now (indirectly via city C).

4. Translation

*“It is ironic how we have computers and phones, yet we still need human translators.” - **Stroufucious***

Problem Statement:
Despite practically owning Stroudonia, The King of Aruba doesn’t know a lick of Stroudonian. However, he needs to speak about policy matters with the Stroudonians. Can you help him translate?

Given a word x in Stroudonian (only containing letters in the English alphabet), the steps for translation are as follows-

- First, perform a cyclic shift[(1)] 67 places. If $|x|$[(2)] is even, you will perform a cyclic shift left (so the first character will become the last character). Otherwise, you will perform a cyclic shift right (the last character will become the last character).
- Next, let m be the median ASCII value of all the characters in x. Then, for each character i, remove i from x if the ASCII value of i is strictly smaller than m. Note that this step <u>may</u> change $|x|$.
- Then, invert the cases for every character in x (all uppercase become lowercase, and vice versa).
- Lastly, for each character i, change it to the letter $|x|$ place(s) after i. For example, if $|x|$ is 3, a ‘b’ will be changed to an ‘e’. Shifts that go past ‘z’ (or ‘Z’ for uppercase letters) will not be performed (e.g. shifting y by 3 will not be done, and the original character will be retained).
- The resulting string is the complete translation from Stroudonian to Aruban.

1- A cyclic shift right of i moves all characters right i places, wrapping the last i characters back around to the start (ex: “abcd” right 2 ⇒ “cdab”). Similarly, a cyclic shift left of i places moves all characters left i places, wrapping the first i character back around to the end.

2- $|x|$ denotes the length of string x.

Input(input arrives from stdin):
The first line of input contains T, the number of test cases. Each test case will be formatted as follows-

The first line will contain N - the length of the string.

The second line will contain the string x - the string to be translated. It is guaranteed that x will contain only lowercase and uppercase characters found in the English alphabet.

CONTINUED ON NEXT PAGE

Translation(Cont.)

Output(output to stdout):

For each test case, output the translated string.

Restrictions:

- $(1 \le T \le 10)$, $(1 \le N \le 1000)$

Sample Input:

```
3
5
Hello
6
SLHSCS
6
Carter
```

Sample Output:

```
ORO
vvv
UWU
```

Explanation:

In the first test case has the following steps for translation–

1. Since $|x|$ is odd, we will shift x left 67 places. The resulting string will be "loHel".
2. The median value of the ASCII values of every character in x 108. Therefore, we will remove 'H' and 'e' since their ASCII values are 72 and 101, respectively. The resulting string will be "lol".
3. Invert the cases of every character. The resulting string will be "LOL".
4. Shift every character in x forward in the alphabet by $|x| = 3$. 'L' will become 'O' and 'O' will become 'R'. Thus, the final string/answer will be "ORO".

5. Election

16 A.S. - The Arubans have finally finished their business and are allowing fair elections. However, Mr. Melon and his cronies have other ideas.

Problem Statement:
After 16 years of Aruban rule, the election will finally be held. As Mr. Melon's campaign advisor, you have been tasked with bribing election officials to guarantee that he wins in the upcoming elections. However, he only has K Stroudollars to do so, funded by a shadowy cabal of tech companies. The election will have C candidates. Conveniently, each candidate was given a number 1 through C, with Mr.Melon being the Lth candidate. Furthermore, candidate $i(1 \le i \le C)$(the candidate given the ith number) currently has a_i votes.

The election is structured as a **tree** with N nodes. A **tree** is a connected graph without cycles. Every node of the **tree** represents a round where candidates compete, and the winner at each node is the **median** out of all of its children. The **median** of a sorted list with n elements is the item at position $n/2$ if n is even and $(n + 1)/2$ if n is odd. Initially, all candidates will start at the leaves, and the winner of each round will progress up the tree by ascending to the parent node. Once the election process has concluded, the overall winner will be the candidate at node 1. For 1 Stroudollar, you may bribe the election authorities to change the reported number of votes for any candidate except Mr. Melon by 1 point. Determine if it's possible to guarantee that Mr. Melon wins while staying within the budget.

Input Format(input arrives from stdin):
Line 1: C, N, K, L — number of contestants, nodes, budget, and the index of Mr. Melon.
Line 2: $a_1, a_2, \cdots, a_C$ — the number of votes each contestant has.
Line 3: $l_1, l_2, \cdots, l_C$ where l_i represents the node where candidate i is located at the start(they will all be leaf nodes and all leaf nodes will be filled).
Line 4: $p_2, p_2, \cdots, p_N$ — the parent of each node (node 1 is the root, so it isn't included). It is guaranteed that there are exactly C leaf nodes.
Output Format(output to stdout): Output a single line containing either "YES" if it is possible that Mr. Melon wins or "NO" if not.
Restrictions:

- $(1 \le C \le N \le 3000)$, $(1 \le a_i \le 10^5)$, $(1 \le l_i \le N)$, $(1 \le p_i \le N)$

Sample Input 1:

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

CONTINUED ON NEXT PAGE

Election(Cont.)

Sample Output 1:

```
NO
```

Sample Input 2:

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

Sample Output 2:

```
YES
```

Sample Input 3:

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

Sample Output 3:

```
YES
```

Explanation:

In the first sample, Mr. Melon needs 0 votes to win, which is unattainable with a budget of 1 Stroudollar(the budget needs to be 2 since he has 2 votes).

For the second sample, Bribe the officials to reduce the number of votes candidate 2 has from 3 to 2 (cost: 1 Stroudollar). New vote tallies: [2, 2, 1, 4]. In the first round, Candidates 1(Mr.Melon) and 2 face off to determine the winner at node 2. Both of the candidates have the same number of votes, so the tie would be broken by index. Sorted list of the two candidates: [Mr. Melon, Candidate 2].Since the length of the array is 2, the winning candidate is at position 2/2 = 1 → Mr. Melon wins and moves on to the next round
In the final round, Candidates 1, 3, and 4 compute with vote tallies of 2, 1, and 4, respectively. Sorted list of all of the candidates: [Candidate 3, Mr. Melon, Candidate 4], vote tallies in order: [1, 2, 4]. The candidate with the median value is the candidate at position $\lceil 3/2 \rceil$ = 2, which is Mr.Melon with a value of 2 → Mr. Melon wins.

In the third testcase, one way to guarantee that Mr. Melon wins is to lower Candidate 4's vote tally by 1 and increase Mr. Melon's vote tally by one. As a result, the new vote tallies become [5, 3, 4, 3, 4]. Doing this only takes 2 Stroudollars, meeting the budget requirement.

6. Self-Driving Disaster

20 A.S.: Mr. Melon has promised the public “safe fully driverless” cars to boost sales for the shadowy cabal of companies that got him elected. However, because of the lack of oversight over the testing of the cars they aren’t safe In fact, you could say it was trying to make the cars crash…

Problem Statement:
Currently, there are N self driving cars that travel in a straight line going either West(“W”) or South(“S”) at a rate of one unit per second. Each car will start at position (x_i, y_i).

A collision occurs if and only if two cars occupy the same point **at the same time**. The AI has found a way to make two cars disappear after colliding, leaving no trace of their existence. With this in mind, answer Q queries in the form: how many collisions occurred before or at time t?

Input Format(input arrives from stdin):
The first line of input contains N and Q, the number of cars and queries respectively.
Each of the next lines will contain the direction d_i represented by “W” or “S” as well as x_i and y_i, the coordinates of the ith car. No two start at the same position.
The jth of the next Q lines will contain a query time t_j.

Output Format(output to stdout):
Output Q lines answering all of the queries.

Restrictions:

- $(1 \le N \le 2 \cdot 10^5)$, $(1 \le Q \le 2 \cdot 10^5)$, $(1 \le t \le 10^9)$, $(-10^9 \le x_i, y_i \le 10^9)$

Sample Input:
```
4 1
W 3 3
S 2 4
W 4 3
S 0 7
4
```

Sample Output:
```
2
```

Explanation: Cars 1 and 2 will collide at position (2, 3) at time t = 1, causing them to disappear afterwards. Then, cars 3 and 4 will collide at position (0, 3) at time t = 4. Thus, there will be 2 collisions.

7. Takeover

15 A.S. - Mr. Melon's lax regulations on the tech industry have backfired as the newly created robot assistants have decided to take power for themselves.

Problem Statement:
The robots of Neo-Stroudonia are planning an attack using blueprints of the city. Each blueprint is a rectangular grid of symbols, where each symbol has an associated strength equal to its ASCII value.

- **Scale Up:** Each cell in the blueprint is repeated in a $K \times K$ block.
- **Divide Into Chunks:** The scaled-up blueprint is divided evenly into R rows and C columns of chunks. Each chunk has dimensions (H*K)/R × (W*K)/C. You are guaranteed that (H*K) is divisible by R and (W*K) is divisible by C.
- **Identify Strongest Chunk(s):** Compute the sum of ASCII values of all cells in each chunk. The chunk(s) with the highest sum are the highest priority. If multiple chunks tie, all are selected.
- **Return the final map.**

Input Format(input arrives from stdin):
The first line of input contains H and W, the rows and columns of the plans.
The next H lines will contain the floor plans in their entirety.
The next line contains an integer K, the scale.
The last line contains two integers R and C, the number of chunk rows and columns.
Output Format(output to stdout):
Print the scaled-up blueprint after destroying the strongest chunk(s).
Restrictions:

- $(1 \leq H, W \leq 20)$, $(1 \leq K \leq 5)$, $(1 \leq R, C \leq 10)$

Sample Input:
```
3 3
*$A
@!1
P:-
2
3 3
```

Sample Output:
```
**$$AA
**$$AA
@@!!11
@@!!11
..::--
..::--
```

CONTINUED ON NEXT PAGE

Takeover(Cont.)

Explanation:
The original blueprint is 3×3, scaling by K=2 makes it 6×6. Splitting the grid into R=3 rows and C=3 columns makes 9 chunks.
Summing the ASCII values of each chunk shows the bottom left chunk is the strongest. The bottom left chunk is replaced with “.”s.

8. Last Stand

*Urgent transmission to Aruba- “We are under heavy attacks from the product of our own imagination, please sen… reinfo… imedia…” *message is cut out* .*

Problem Statement:
The robots are coming! The officials of the Stoudonian government and Mr. Melon have retreated to the capital building and are currently being pinned down by waves of RoboFighter 1.0s. More specifically, the group will initially face an attack of power level N on day 1, and they will face subsequent attacks every X days. The defense group’s initial power level is K, and they will successfully defend an attack only if their current power level is strictly greater than the attack’s power level. However, the robots are very smart, so every time their attack fails, they will double their power level for the next attack (i.e. attack number A will have a power level of $N*2^{(A-1)}$). Additionally, each day that the group is not facing an attack, they will gain J power levels. Can you help the group figure out the day of the first attack the defense group will not be able to defend?

Input(input arrives from stdin):
The first line of input contains T, the number of test cases. Each test case will be formatted as follows-

The first and only line will contain N, X, K, and J - the power level of the initial attack, the frequency of the attack, the defense group’s initial power, and the amount of power levels gained for each day without an attack, respectively.
Restrictions:

- $(1 \leq T \leq 10)$, $(1 \leq N \leq 100)$, $(1 \leq X \leq 10)$, $(1 \leq K \leq 10^5)$, $(1 \leq J \leq 100)$

Output(output to stdout):
For each test case, output one integer- the day of the first attack the defense group will not be able to defend.
Sample Input:
```
3
2 2 5 1
5 1 4 1
2 1 8 10
```

Sample Output:
```
5
1
3
```

CONTINUED ON NEXT PAGE

Last Stand(Cont.)

Explanation:
In the first test case, each day will have the following results-

Day 1 (attack)- Attack power: 2, Defense power: 5, Defense is successful.
Day 2 (no attack)- Initial defense power: 5, power levels gained: 1, new defense power: 6.
Day 3 (attack)- Attack power: 4, Defense power: 6, Defense is successful.
Day 4 (no attack)- Initial defense power: 6, power levels gained: 1, new defense power: 7.
Day 5 (attack)- Attack power: 8, Defense power: 7, Defense is <u>not</u> successful.

Since they cannot defend the first attack they cannot defend is on day 5, the answer is 5.

In the second test case, they will not be able to defend the attack on day 1 as the initial defense level is lower than the initial attack level.

9. “Rehabilitation”

DIARY ENTRY 01 22 A.S.: We live in constant fear of being “rehabilitated”. If the robots don’t like our work, they will simply discard us.

Problem Statement:
The robots would like to measure the performance of their “valued contractors” so they can decide whether to “rehabilitate” the workforce. They have N “contractors”, the ith of which has a skill score of a_i.

The **effectiveness** of the workforce is the number of pairs of integers (l, r) such that the sum of elements $a_l, a_{l+1}, ..., a_r$ is equal to $r - l + 1$. If the **effectiveness** of the workforce is strictly less than their threshold K, then they will “rehabilitate” everyone! Determine whether they will do so!

Input(input arrives from stdin):
The first line of input will contain three integers N, K, the number of workers and the robots’ effectiveness threshold, respectively. The second line of input will contain N integers $a_1, a_2, ..., a_N$, skill values of the workers.

Output(output to stdout):
Output “All is well” if the robots won’t “rehabilitate” the workforce and “ESCAPE BEFORE THEY REHABILITATE US!!!” otherwise.

Restrictions:

- $(1 \leq N, K \leq 2 \cdot 10^3)$, $(-10^9 \leq a_i \leq 10^9)$

Sample Input 1:

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

Sample Output 1:

```
All is well
```

Sample Input 2:

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

Sample Output 2:

```
ESCAPE BEFORE THEY REHABILITATE US!!!
```

Explanation:
In the first test case, subarrays [1, 4], [3, 3], and [6, 6] with sums of 4, 1, and 1 satisfy the conditions. Since this is ⩾ 3, output is “All is well”. In the second test case, there are only 3 subarrays, meaning that it isn’t valid.

10. Famine

DIARY ENTRY 02 29 A.S.: Survival is a daily struggle. Food is scarce. The robots have no use for us except for harvesting energy.

Problem Statement:
Following the shift to modern technology, many have lost their jobs, and food has become scarce. As a result, Jeff wants to know many more days he will still have on this barren wasteland. More specifically, Jeff has N Stroudollars, and the supermarket has M items for sale. Each item will cost C_i Stroudollars and give H_i hunger levels to Jeff; the supermarket will have S_i of that item in stock at a time, meaning that Jeff can only buy up to S_i of the said item. However, the supermarket will also restock every R days, resetting the stock of each item to the original S_i on day 0.

Jeff will go to the supermarket at the beginning of each day, and his hunger level will decrease by 1 at the end of the day (meaning that if he starts off a certain day with a hunger level of 0, he can survive that day as long as he has a hunger level $\geqslant 1$ by the end of the day).

Jeff wants to know how long he can survive for a given scenario. Please help him with this task! Jeff will die if his hunger level is smaller than or equal to 0 at the end of a day.

Input(input arrives from stdin):
The first line of input contains T, the number of test cases. Each test case will be formatted as follows-

The first line will contain N, M, and R - the amount of money Jeff starts with, the number of items in the supermarket, and restock rate of the supermarket.

The next M lines will contain the items in the supermarket. Line $i + 1$ will contain N_i (guaranteed to be one word), C_i, H_i, and S_i - the name, cost, hunger level, and stock of item i, respectively.
Output(output to stdout):
For each test case, output a single integer - the maximum number of days Jeff can survive for given the current scenario.
Restrictions:

- $(1 \leq T \leq 10)$, $(1 \leq N \leq 10^9)$, $(1 \leq M \leq 100)$, $(1 \leq R \leq 10^5)$, $(1 \leq S_i \leq 10)$

CONTINUED ON THE NEXT PAGE

Famine(Cont.)

Sample Input:

```
3
100 2 20
Bread 5 3 3
Cupcakes 2 1 1
5 1 1
EnergyBar 1 1 1
1 1 1
Cookies 2 2 2
```

Sample Output:

```
10
5
0
```

Explanation:
In the first test case, Jeff will buy every item in the supermarket on day 0 (3 Bread and 1 Cupcake for a total of 15+2=17 Stroudollars). The Bread will give him a total of 3*3=9 hunger levels and the Cupcakes will give him a total of 1*1=1 hunger levels. Therefore, Jeff can survive for a total of 9+1=10 days. It can be shown that this is the maximum number of days he can survive for.

In the second test case, Jeff will buy Energy Bar everyday for 5 days (when he runs out of money). He can buy the Energy Bar 5 times (despite the stock being 1) because the supermarket will restock all the items in the store everyday. As a result, Jeff will be able to live for 5 days.

11. Recruitment

DIARY ENTRY 002 39 A.S.: I heard on the streets that our glorious leader Stroud has returned and started a rebellion!

Problem Statement:
There are N different advertisements that need to be analyzed. Each advertisement has the following information associated with it:

- The number of people that the advertisement was served to.
- The cost of serving each person the advertisement, in Stroudollars($).
- The position the advertisement was about. For example, an advertisement could advertise being a drone pilot or a janitor.
- The portion of people that were interested in the advertisement(the percent can be anything because math works differently in Stroudonia).

It is your job to output a formatted table containing the following data points in the same order:

- The role that was advertised.
- The <u>number</u> of people that were interested in an advertisement.
- The engagement ratio of each advertisement(# of engaged people/cost of the ad).
- The total cost of each advertisement(this will never be greater than $999.99).
- Percent of the total cost of all advertisements spent on this one.

Input(input arrives from stdin):
The first line of input will contain N, the number of advertisements.

Each of the next N lines will contain information about each advertisement separated by commas: the advertised position, the per person cost, the number of people that it was served to, and the portion of people that were interested.

Output(output to stdout):
Output a formatted table containing the information specified.
Note that money and percentages should be outputted to two decimal places. Put a $ before values representing money and a % after percentages.

Restrictions:

- $1 \leq N \leq 100$

Sample Input 1

```
4
Janitor, 0.01, 50, 20.00
Drone Pilot, 2.00, 20, 100.00
Foot Soldier, 30.00, 1, 1.01
Chef, 0.51, 1000, 89.87
```

CONTINUED ON THE NEXT PAGE

Recruitment(Cont.)

Sample Output 1:

```
| Position     | # of People | Engagement | Cost    | %of Total |
| ------------ | ----------- | ---------- | ------- | --------- |
| Janitor      | 50          | 20.00      | $0.50   | 0.09%     |
| Drone Pilot  | 20          | 0.50       | $40.00  | 6.89%     |
| Foot Soldier | 1           | 0.00       | $30.00  | 5.17%     |
| Chef         | 1000        | 1.76       | $510.00 | 87.86%    |
```

Sample Input 2:

```
2
Writer, 0.32, 12, 12.29
Intern, 3.10, 3, 32.10
```

Sample Output 2:

```
| Position | # of People | Engagement | Cost  | %of Total |
| -------- | ----------- | ---------- | ----- | --------- |
| Writer   | 12          | 0.38       | $3.84 | 29.22%    |
| Intern   | 3           | 0.10       | $9.30 | 70.78%    |
```

Note:
The size of the table changes from test case to test case because the longest value changes. For example, the first column in sample 1 is longer than the one in sample 2 because the longest value(Foot Soldier) is longer than the longest value in sample 2(the column name, Position). There should always be one space separating the entry and the “|”.

12. Is It Enough?

Head Admiral Dr. Nutt: *"Do we have operatives to actually get this rebellion going?!?!!? We should have gotten enough by now, shouldn't we? Jeff!!! I need you to solve this problem for me!"*

Problem Statement:
The Head Admiral would like to know if the rebellion has enough operatives to set their plans in motion. Currently, the rebellion has N people and it needs X operatives to get started. Since the Admiral is too stupid to compare two numbers, he has tasked you with answering this question.

If the rebellion has more than enough operatives, output "WE HAVE ENOUGH SIR!". Otherwise, output "WE NEED K MORE OPERATIVES SIR!" where K is the number of operatives that the rebellion is lacking.

Input Format(input arrives from stdin):
There will be only one line of input containing two integers: N and X, representing the number of recruits the rebellion has and needs, respectively.
Output Format(output to stdout):
Output "WE HAVE ENOUGH SIR!". Otherwise, output "WE NEED X MORE OPERATIVES SIR!" where X is the number of operatives that the rebellion is lacking.
Restrictions:

- $(1 \le N \le 10^{18})$, $(1 \le X \le 10^{18})$

Sample Input 1
```
3 2
```

Sample Output 1
```
WE HAVE ENOUGH SIR!
```

Sample Input 2
```
3000 3353
```

Sample Output 2
```
WE NEED 353 MORE OPERATIVES SIR!
```

Explanation:
In the first testcase, the rebellion already had 3 operatives when they only needed 2.

In the second testacse, the rebellion was short 3353-3000 = 353 operatives.

13. Communication Network

“Just like a graph, a good rebellion is a completely connected one.”

-Our Glorious Leader Stroud

Problem Statement:
The rebellion has N field operatives that need to be connected to the comms network. If an operative located at position (a, b) and the base is located at (c, d), it takes $|a - c| + |b - d|$ Stroudollars to connect them. With this in mind, find the minimum cost to connect all of the operatives to the network given an optimal selection of the location of the base.

Input Format(input arrives from stdin):
The first line of input will contain one integer N, the number of operatives.

Each of the next N lines of input will contain two integers: x_i and y_i, representing the coordinates of the ith operative.

Output Format(stdout):
Output the minimum total cost it takes to create the communication network given optimal placement of the base.

Restrictions:

- $(1 \le N \le 2 \cdot 10^5)$, $(-10^9 \le x_i, y_i \le 10^9)$

Sample Input 1:
```
2
-1 1
1 3
```

Sample Output 1:
```
4
```

Sample Input 2:
```
5
1 2
3 3
1 5
2 -2
-1 3
```

Sample Output 2:
```
13
```

Explanation:
In the first test case, one of the best positions for the base is at (0, 2).
The cost for this configuration is |0-(-1)| + |0-1| + |2-1| + |2-3| = $4.
In the second test case, the base can be placed at (1, 2) to get $13.

14. Message Interception

HQ MESSAGE 003, 44 A.S.:
ROBOT MESSAGES INTERCEPTED. SEND MOST IMPORTANT MESSAGES ASAP SO WE CAN FIND THE LOCATION OF THE MAINFRAME. AWAIT FURTHER INSTRUCTIONS.

HQ has intercepted encrypted messages between robot officials, T of them to be exact. Each message is a string S of length N consisting of digits between 0 and 9. The **importance** of a message is defined as the number of positions $i (2 \le i \le N - 1)$ where $S_i > S_{i-1}$ and $S_i > S_{i+1}$. HQ needs you to return the T messages sorted in terms of their **importance** so they can optimize the decoding process. If two messages have the same importance, the one that appeared earlier in the input should come first in the output! Good luck soldier!

Input Format(input arrives from stdin):
The first line of input will contain two integers: T and N, the number of intercepted messages and the length of each message.

Each of the next T lines of input will contain a string S consisting of digits in "$0 \ldots 9$".
Output Format(output to stdout):
Output the T messages sorted in terms of their **importance**.
Restrictions:
- $1 \le T \le 10$, $1 \le N \le 100$

Sample Input:
```
3 5
19022
90909
11111
```

Sample Output:
```
19022
90909
11111
```

Explanation:
19022 has an **importance** value of 1 since the only position that satisfies the condition is position 2(9 > 0, 9 > 1).
90909 has an **importance** value of 1.
Finally, 11111 has an **importance** of 0.
The sorted list will be [19022, 90909, 11111]. Even though 19022 and 90909 have the same **importance**, 19022 is placed before 90909 in the sorted list because it was intercepted first.

15. Job Assignment

45 A.S. - **Speech by our Glorious Stroud:** *"Soldiers! Fulfill your assignments well! Today will be our historic reconquest of Stroudonia!"*

Problem Statement:
There are N operatives and M jobs that they need to be assigned to. You will be provided the following information about each operative:

- Their first and last name: It will consist of no more than 20 upper case and lower case latin letters("a … zA … Z") separated by a space.
- The jobs that they are most skilled at. The operative must be assigned to one of these given jobs.
- Their friendliness, which is a real number between 0 and 1, inclusive(values have only 2 decimal places)

All of the operatives will be assigned to teams with the same job. In order to ensure the greatest efficiency of the teams, the following constraints must be satisfied:

- Each person must be assigned to one of the jobs they are skilled at.
- Average friendliness of each job team must be strictly greater than 0.5.
- No more than 3 people with the same first name in a team.
- No two people in the same team can have the same last name.
- Operatives named **Jeff** must be assigned to **MainframeDestruction**. If Jeff can't be assigned to **MainframeDestruction**, no valid assignment exists.
- Job assignments must respect the lexicographic order of operatives:
    - Sort operatives by (first name, last name)
    - Let job(i) be the index of the job assigned to the i-th operative in this order. Then, job(1) ≤ job(2) ≤ ... ≤ job(N)

Given these constraints, produce **ANY** feasible assignment where each person is assigned to exactly one job or report that it is impossible. Note that a job may have no people assigned to it and all teams have unlimited capacity.

Input(input arrives from stdin):
The first line of input will contain two integers: N and M, the number of operatives and the number of jobs, respectively.

The next M lines contain a list of all of the available jobs. Each job will be a string of less than 20 latin letters in any case. No two jobs will have the same name.

Each of the next N lines will be formatted as follows:
(First Name) (Last Name), (friendliness), Job: (job1), (job2), …
It is guaranteed that no job is repeated twice and that no two operatives have the same first name and last name combo.

CONTINUED ON THE NEXT PAGE

Job Assignment(Cont.)

Output(output to stdout):
Output M lines, one per job, in the same order they were given in the input. Each line should contain the job name followed by a “:” and a comma and single space separated list(“, “) of the people assigned to the job. Also, all of the names must be outputted in the lexicographic order described above.

If no valid assignment exists, simply output “Everyone for themselves!”
Restrictions:

- $(1 \leq N, M \leq 20)$

Sample Input 1:
```
3 3
MainframeDestruction
Cooking
CentralCommand
Jeff Luap, 1.00, Jobs: MainframeDestruction
Melon Moosk, 0.10, Jobs: MainframeDestruction, Cooking
Austin Yum, 0.01, Jobs: CentralCommand, Cooking
```

Sample Output 1:
```
Everyone for themselves!
```

Sample Input 2:
```
4 5
Nothing
Eating
MainframeDestruction
ShipRepair
BeingAPirate
Jeff Stephens, 0.60, Jobs: MainframeDestruction, Eating
Eren Tor, 1.00, Jobs: ShipRepair, BeingAPirate, Nothing, Eating
Abhi Good, 0.30, Jobs: Eating, Nothing
Jeff Luap, 1.00, Jobs: MainframeDestruction, ShipRepair
```

Sample Output 2:
```
Nothing:
Eating: Abhi Good, Eren Tor
MainframeDestruction: Jeff Luap, Jeff Stephens
ShipRepair:
BeingAPirate:
```

16. Navigation

COMMS LOG 0105.01.01:
I need help with navigating my way to the robots' mainframe since I've never been in Neo-Stroudonia before. Can you send me the fastest route to the mainframe? Curfew is coming soon so please hurry.

Problem Statement:
Neo-Stroudonia's capital city, Luap-land, has many winding streets, dead-ends, and loops that can easily confuse even the most experienced resident.. Through this chaos, you must navigate to the location of the citadel that houses the mainframe.

It can be modeled by a NxM coordinate grid with each square labeled as follows:

- 'S' - The starting position.
- 'E' - The location of the citadel that houses the mainframe. This is the square you are trying to get to.
- '.' - A street where you can move.
- '#' - A wall that you can't cross.
- 'J' - A jump pad. It will allow you to jump over a wall 1 unit away, thereby going 2 units in any direction(there doesn't need to be a wall).

Write a program that finds the fastest route to the citadel ASAP!

Input Format(input arrives from stdin):
The first line of input will contain two integers N and M, representing the dimensions of the grid.

Each of the next N lines will contain M characters that represent the information regarding each cell of the grid.
Output Format(output to stdout):
Output the minimum amount of time it takes to reach the citadel or report that it is impossible by outputting "I can't make it. I'm sorry guys!"
Restrictions:

- $(1 \leq N, M \leq 100)$

Sample Input 1:
```
4 3
.#.
.#S
.#J
E.#
```

Sample Output 1:
```
3
```

CONTINUED ON THE NEXT PAGE

Navigation(Cont.)

Sample Input 2:

```
5 6
S#.#..
...#..
.J...#
####..
E.....
```

Sample Output 2:

```
5
```

Explanation:

In the first test case, the best path is to start at the ‘S’, go down one to the jump boost, jump downwards over the wall, and head straight for the Exit(‘E’). In total, this takes 3 seconds.

In the second test case, the most optimal path is to head straight towards the jump pad, ‘J’, in 4 seconds and then jump over the wall to reach the citadel in 5 seconds.

17. Password

COMM LOG 10 0102.09.14: Haev arved at citadel. Robots have unxpctdly put a lo9lck on the door. Nneed help asap.

Problem Statement:
The password is a string consisting of “?”s, 1s, and 0s. However,the robots wanted to make the password long, so they took a base string S of length N and concatenated it to itself K times. For example, the string “101?” repeated 3 times would be “101?|101?|101?”(“|”s added for clarity). Robots absolutely hate **mistakes**. A **mistake** occurs when a 1 is before a 0(e.x. “100” has 2 mistakes). Note that **mistakes** can occur between any two positions in the full concatenated string, including positions from different copies of S. Because of this, they tried to minimize the number of **mistakes** by secretly changing all of the “?”s in S to either 1s and 0s **before** concatenation. Find the minimum number of **mistakes** that can occur after optimal replacement of all “?”s, modulo(remainder) **998244353**.

Input(input arrives from stdin):
The first line of input will contain two integers: N and K, the length of S and the number of times it is concatenated to itself.
The second line of input will contain a string S, the basis string.
Output(output to stdout):
Output the minimum number of **mistakes**, remainder **998244353**.
Restrictions:

- $(1 \leq N \leq 2 \cdot 10^3)$, $(1 \leq K \leq 10^9)$

Sample Input 1:

```
2 2
1?
```

Sample Output 1:

```
0
```

Sample Input 2:

```
5 3
10?1?
```

Sample Output 2:

```
15
```

Explanation:
Sample 1: The only ? can be set to 1. String S becomes “11” and the password is “11|11”. The number of **mistakes** is 0.
Sample 2: String S can be changed to “10111”. The password becomes “10111|10111|10111”. Here, there are 15 **mistakes**. Example: positions 1 and 2.

18. Mainframe Destruction

“So this is the mainframe? The thing that controls all of the robots? Pretty impressive, I must say, but today I, Jeff, will DESTROY it!”

Problem Statement:
The mainframe has two boards, A and B, each with N ports. Port i on A connects to port i on B, with charges of a_i and b_i, respectively(i.e. a_1 is connected to b_1, a_2 is connected to b_2, etc.). To make the mainframe **short circuit**, every connected pair of ports must have the same charge.

However, because the robots are clever, they made it so that you can only modify the charges by applying the following operation any number of times:

- Pick a number $i (1 \le i \le N - 1)$. Simultaneously set $a_i = a_{i+1} - 1$ and $a_{i+1} = a_i + 1$.
- For example, the charge array [1, 10, 100] becomes [9, 2, 100] after applying an operation with $i = 1$.

It is your task to figure out the **minimum number** of operations needed to make the mainframe **short circuit**.

Input Format(input arrives from stdin):
The first line of input will contain one integer N, representing the number ports each circuit board has.

The second line of input will contain N integers $a_1, a_2, \cdots, a_N$, representing the charges for each port in circuit board A.

The third line of input will also contain N integers $b_1, b_2, \cdots, b_N$, representing the labels for each port in the second circuit board.
Output Format(output to stdout):
Output the **minimum number** of operations required to make the mainframe **short circuit**. If it is not possible to do so, output -1.
Restrictions:

- $(1 \le N \le 2 \cdot 10^5)$, $(1 \le a_i, b_i \le 2 \cdot 10^5)$

Sample Input:
```
4
3 2 4 5
3 3 4 4
```

Sample Output:
```
2
```

CONTINUED ON THE NEXT PAGE

Mainframe Destruction(Cont.)

Explanation:

First, the operation can be applied to $i = 2$, making the charges in circuit board A = [3, 3, 3, 5]. After this, the operation can be applied to $i = 3$, resulting in the charges of A being [3, 3, 4, 4]. Since this matches the charges of circuit board B, it only takes 2 operations to make the mainframe short circuit.