# 2018 KAIST RUN Spring Contest 

## Sponsored By:

## ZOYI



Startlink

## STARTLIN<

## Rules

- This contest is 2018 KAIST RUN Spring Contest
- This contest is sponsored by ZOYI and Startlink.
- This contest starts from 13:00 to 18:00, May 22nd 2018, Korean Time (GMT +9).
- You can only participate individually.
- Use of the network is prohibited during the competition, except for submitting source codes and accessing language reference sites. Here are examples of allowed reference sites.

```
- C/C++ : http://en.cppreference.com/w/
- Java : https://docs.oracle.com/javase/8/docs/api/
- Python: https://docs.python.org/
- Kotlin: https://kotlinlang.org/docs/reference/
```

- Using any pre-written code or prints before contest is NOT allowed.
- This contest consist of 11 problems.
- Problem is NOT sorted by difficulty.
- Every problem is guaranteed to solvable using $\mathrm{C}++17$ in time limit and memory limit.
- Each problem consists of 1 or more subtasks, and each subtask is worth certain number of points. Subtask means easier problem with possible additional constraints.
- Each subtask may have additional constraints. Full constraints of subproblems includes constraints of full problem and subtasks' constraints.
- Each subtask consists of several test data, and points for the subtask are awarded if every test data is passed. The total number of points for each problem is 100 points.
- Every problem have time limit and memory limit. This means your problem should run in given time and memory for each data. Memory limit includes stack memory and heap memory.
- Memory limit for every problem is fixed as follows.:
- C11 / C++17 : 1024MB
- Java / Kotlin : 1536MB
- Python 2 / Python 3: 2080MB
- PyPy2 / PyPy3 : 2176MB
- Each problem may have different time limit.
- Each participant's ranking is determined in the following way::
- Penalty time $=$ (Duration from contest start to last submission that increased points)
- Ranking $=(\#$ of participants with higher points $)+(\#$ of participants with same points and lower penalty time) +1


## Problem list

| $\#$ | Problem Name | Time limit | Score | Subtask score |  |  |  |  |
| :---: | :--- | ---: | :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | (All language) | Full score | 1 | 2 | 3 | 4 | 5 |
| P | Puyo Puyo | 1 second | 100 | 63 | 37 |  |  |  |
| Q | QueryreuQ | 1 second | 100 | 36 | 64 |  |  |  |
| R | Recipe | 1 second | 100 | 17 | 20 | 63 |  |  |
| S | Segmentation | 1 second | 100 | 30 | 70 |  |  |  |
| T | Touch The Sky | 1 second | 100 | 22 | 33 | 45 |  |  |
| U | United States Of Eurasia | 20 seconds | 100 | 20 | 20 | 60 |  |  |
| V | Voronoi Diagram | 1 second | 100 | 27 | 73 |  |  |  |
| W | Winter Olympic Games | 5 seconds | 100 | 23 | 24 | 53 |  |  |
| X | Xtreme NP-hard Problem?! | 5 seconds | 100 | 19 | 27 | 54 |  |  |
| Y | Yut Nori | 1 second | 100 | 31 | 13 | 10 | 27 | 19 |
| Z | Zigzag | 1 second | 100 | 33 | 37 | 30 |  |  |

## Problem P. PuyoPuyo

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
1 second
1024 megabytes


Figure: Screenshot of PuyoPuyo game. This game uses $12 \times 6$ grid. There are 4 kinds of color.
PuyoPuyo is a famous series video game made by Compile Co,. Ltd.. First game was released in 1991. The company defunct at 2003, but PuyoPuyo team is still making game in Sonic Team ${ }^{\mathrm{TM}}$ to this day, 2018.

PuyoPuyo is a 2 player battle game. Each player plays by placing Puyo to grid. The detailed game rule is as follows:

- The game starts with an empty grid.
- Puyo is a round, slime-like object that falls from above to below of the grid.
- Each Puyo has a color.
- There are $K$ kinds of color of Puyo.
- Puyos are controlled as a pair (two Puyos).
- Puyo pair can be moved, dropped or rotated using a game controller.
- At the topmost part of the screen, Puyo pair can be moved horizontally, and can be rotated in horizontal or vertical direction.
- When you drop the Puyo, it each Puyo drops separately, until it reaches another Puyo or the bottom of the grid.
- You can place Puyo outside of the grid, but only over the grid.
- More than four Puyos that have same colors and shares an edge is called Group, and such groups are removed. This process is called Popping.
- If dropping a Puyo pair makes two or more groups, it is popped simultaneously.
- After group pops, remaining Puyos fall to another Puyo or the bottom of the grid. If it makes a new group, same procedure is repeated. This process is called Chain.
- During chains, you cannot place another Puyo pair.

Sonic Team ${ }^{\text {TM }}$ requested your help to make PPAP (Puyo Puyo Algorithm for Printing). This will not be used in game software but will be used in special ceremonies.
You are given $R \times C$ grid. Each grid is filled with certain color of Puyo or an empty grid. You should drop Puyo pair and derive a given grid. In the final state, every Puyo should be in the grid.

Fortunately, you can choose which color and place to drop.

## Input

Input consists of $R+1$ lines.
First line contains three space-separated integers $R, C$ and $K$.
Next $R$ lines contain the final grid. Each line contains $C$ number of space-separated integers. Each color is represented by an integer 1 through $K .0$ means an empty grid.
A given final grid can be made by no more than 250 Puyo pair drops.

## Output

In the first line, you have to print the number of Puyo pair $D$.
In following $D$ lines, print four space-separated integers in each line.

- If you place Puyo pair horizontally, the first number is 0 . If you place vertically, the first number is 1.
- Second number represents the row number of the left Puyo.
- Third number represents the color of left or upper Puyo of the pair.
- Fourth number represents the color of right or lower Puyo of the pair.


## Constraints

- $3 \leq K \leq 6$
- $R=C=4$ or $R=C=20$
- $D \leq 250$ (You have to use 250 or less Puyo pairs.)


## Subtask 1 (63 points)

This subtask has additional constraints :

- $R=C=4$

Subtask 2 (37 points)
This subtask has no additional constraints.

## Examples

| standard input | standard output |
| :---: | :---: |
| 443 | 5 |
| 0000 | 0212 |
| 0000 | 1412 |
| 0000 | 0223 |
| 0101 | 1333 |
|  | 1323 |

## Notes

In this example, Puyos are dropped like following :


## Problem Q. QueryreuQ

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
1 seconds
1024 megabytes

A string is palindrome, if the string reads the same backward and forward. For example, strings like "a", "aa", "appa", "queryreuq" are all palindromes.
For given empty string $S$, you should process following two queries :

1. Add a lower case alphabet at the back of $S$.
2. Remove a character at the back of $S$.

After processing a query, you should count the number of palindrome substring in $S$. For string $S$ and integers $i, j(1 \leq i \leq j \leq|S|), S[i, j]$ represents a substring from $i$ th to $j$ th character of $S$. You should print out the number of integer pairs $(i, j)$ where $S[i, j]$ is palindrome.

## Input

Input consists of two lines.
In the first line, $Q$, the number of queries is given.
In the second line, the query is given as string of length $Q$. $i$ th character $K_{i}$ denotes the $i$ th query.
$K_{i}$ is ' - ' or lower case alphabet ('a', 'b', $\cdots$, ' $z$ ') (without quotes).
If the character is ' - ', you should remove a character at the back of $S$. If the character is lower case alphabet, you should add a character $K_{i}$ at the back of $S$.
It is guaranteed that length of $S$ is always positive after the query.

## Output

Print out $Q$ space-separated integers in the first line. $i$-th integer should be the answer of the $i$ th query.

## Constraints

- $1 \leq Q \leq 10,000$


## Subtask 1 (36 points)

This subtask has additional constraints :

- $Q \leq 500$


## Subtask 2 (64 points)

This subtask has no additional constraints.

## Examples

| standard input | standard output |
| :---: | :---: |
| $17$ <br> queryreuq | 12345791113119754321 |

## Problem R. Recipe

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 1024 megabytes
Jaemin likes cooking. He wants to devise several recipes for $N$ days. He devises a recipe in the following order.

1. Buy ingredients at a market and put them in a refrigerator.
2. Think of a recipe.
3. Take out ingredients from the refrigerator and cook.

He can devise a recipe with such a simple way. He wants to cook delicious food as much as possible.
There are new ingredients in the market daily. The ingredients sold on $i$-th day have freshness $F_{i}$. The freshness $F_{i}$ of the ingredients in the refrigerator decreases by 1 everyday. If the ingredients are in the refrigerator, he doesn't buy more ingredients until he cooks with them.
He has cooking skill $C_{i}$ on $i$-th day. His cooking skill advances everyday, so $0<C_{i} \leq C_{j}$ for all $i<j$. If he takes out the ingredients which freshness is $F$ from the refrigerator and cook with cooking skill $C$, a dish with a flavor of $F \times C$ is made. When he cooks, he invites his friend Jaehyun, who is very hygienic, so Jaemin hopes that the ingredients in the refrigerator have freshness greater than or equal to $L_{i}$. If the ingredients don't satisfy the requirement, Jaemin cannot cook that day. Jaehyun's requirement varies everyday, and the requirements for $N$ days are given as $L_{1}, L_{2}, \cdots, L_{N}$.
After he cooks a new dish, he goes to the market the next day to buy new ingredients and think of a new recipe again. Everyday, he may go to the market to buy ingredients, cook, or do nothing for devising a recipe (It is also possible to cook on the day he purchases the ingredients). On the first day, there aren't any ingredients in the refrigerator, he goes to the market to buy some ingredients. On the $N$-th day, he must cook and empty the refrigerator. Let's find the maximum sum of a flavor of the dishes he cooks. If it is impossible to empty the refrigerator on the N-th day because of Jaehyun's particular requirements, print out "Impossible" (without quotes).

## Input

Input consists of four lines.
First line contains $N$.
Second line contains $N$ space-separated integers $F_{1}, F_{2}, \cdots, F_{N}$.
Third line contains $N$ space-separated integers $C_{1}, C_{2}, \cdots, C_{N}$.
Fourth line contains $N$ space-separated integers $L_{1}, L_{2}, \cdots, L_{N}$.

## Output

Print the maximum sum of flavors of the dishes Jaemin cooks. If it is impossible to empty the refrigerator on the $N$-th day, print out "Impossible" (without quotes).

## Constraints

- $2 \leq N \leq 250,000$
- $0<F_{i} \leq 50,000$
- $0<C_{1} \leq \cdots \leq C_{N} \leq 10,000$
- $0 \leq L i \leq 50,000$


## Subtask 1 (17 points)

This subtask has additional constraints :

- $N \leq 5,000$

Subtask 2 (20 points)
This subtask has additional constraints.:

- $L_{i}=0$


## Subtask 3 (63 points)

This subtask has no additional constraints.

## Examples

| standard input | standard output |
| :---: | :---: |
| $\begin{array}{\|llll} \hline 3 & & \\ 10 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 1 & 1 \end{array}$ | 24 |
| $\begin{array}{\|lllll\|} \hline 3 & & & \\ 10 & 1 & 1 & \\ 1 & 2 & 3 & \\ 10 & 10 & 10 \\ \hline \end{array}$ | Impossible |
| ```10 341592 6 5 3 5 10}11112121314151516 17 18 19 141421356 2``` | 526 |

## Problem S. Segmentation

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
1 second
1024 megabytes

ZOYI is developing a tool called Channel which offers a tool to talk with online users in the site. Recently, ZOYI introduced a RF(Recency / Frequency) Model to distinguish users who are using the Channel and decided to classify the users through following calculations.


Figure : Distinguishing users in RF Channel. Horizontal axis represents Recency, while vertical axis represents Frequency.

$$
\left(0<f_{1}<f_{2}<f_{3}<f_{4}, 0<r_{1}<r_{2}<r_{3}<r_{4}, \text { all } f_{i} \text { and } r_{i} \text { are integers. }\right)
$$

$x$ axis represents Recency and $y$ axis represents Frequency. All online users are given values $r, f$ by their connection record, and are classified into one of twelve conditions shown below.

- "New Customer"
- "Promising"
- "About to Sleep"
- "Hibernating"
- "Lost"
- "Potential Loyalist"
- "Need Attention"
- "About to Leave"
- "Champion"
- "Loyal Customer"
- "Can't Lose Them"
- "None"

Among those, "None" means the user has no connection record to the server. If $(r, f)$ is located on two or more classification boundaries, it follows the classification of $(r-0.5, f-0.5)$. For example, if the value of $(r, f)$ is $\left(r_{4}, f_{2}\right)$ it is classified as "Hibernating", while if the value is $\left(r_{3}, f_{4}\right)$, it is classified as "Loyal Customer".
You want to investigate a user status of whom interested in RUN, so you are trying to install the program in the following way:

- $r$ : if the current time is $t, t$ - (most recent access time)
- $f$ : number of visited times

Given events of site users, make a program which classifies the users following the given picture above.

## Input

First line contains four space-separated integers $r_{1}, r_{2}, r_{3}, r_{4}$.
Second line consists four space-separated integers $f_{1}, f_{2}, f_{3}, f_{4}$.
Third line contains a single integer $N$.
Next $N$ lines contains events in time order, where $i$ th element represents the event held at time $i$.
Each event is given as space-separated $A$ and $B$, where $B$ is the username which contains no whitespace with at most 10 alphabets. $A$ has a value of 1 or 2 , where 1 means the user entered the site while 2 means you should print how the user is classified.

## Output

For events which $A$ is 2 , print how the user is classified in each line (without quotes).

## Constraints

- $1 \leq N \leq 100,000$
- $0<r_{1}<r_{2}<r_{3}<r_{4} \leq 10,000$
- $0<f_{1}<f_{2}<f_{3}<f_{4} \leq 10,000$


## Subtask 1 (30 points)

This subtask has additional constraints :

- Every user's length of name is 1 .
- $N \leq 100$


## Subtask 2 (70 points)

This subtask has no additional constraints.

## Examples

| standard input | standard output |
| :---: | :---: |
| $\begin{array}{llll} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 8 & & & \\ 1 & \text { RUN } \\ 1 & \text { Alex } \\ 2 & \text { Alex } \\ 1 & \text { RUN } \\ 1 & \text { RUN } \\ 1 & \text { Alex } \\ 2 & \text { Alex } \\ 2 & \text { RUN } \end{array}$ | New Customer <br> Potential Loyalist <br> Need Attention |

## Notes

The connection status of Alex is $f=1$ (first visit), $r=1$ (time $3-2=1$ ) at time 3. Thus, Alex is classified as "New Customer".

At time 7, the connection status of Alex is $f=2$ (second visit), $r=1$ (time $7-6=1$ ). Thus, Alex is classified as "Potential Loyalist".

At time 8 , the connection status of RUN is $f=3$ (third visit), $r=3$ (time $8-5=3$ ). Thus, RUN is classified as "Need Attention".

Be aware that the given example is not the testcase of subtask 1 .

## Problem T. Touch The Sky

Input file:<br>Output file:<br>standard input<br>standard output<br>Time limit:<br>1 second<br>Memory limit:



Figure: The house floats up in the sky by balloons. This picture is also used in 2018 KAIST RUN Spring Contest poster.

In the year 2117, Professor Jaemin Yu developed a linear-time algorithm for TSP(Traveling Salesperson Problem). Not long after that happened, all computer systems were destroyed, and nuclear weapons demolished all the lands. You, a great computer expert, also lost your job. With a great despair, you lost your meaning of life long ago. All those things that made your heart beat - where had they gone? After questioning yourself again and again, your conclusion is ...

## "If I go to KAIST where I started my first ICPC, can I find a meaning of my life?"

All transportations were destroyed, but you were an avid ICPC participant, and you collected a lot of century-old balloons in Korean Regionals. If you could float a house with some of those balloons...
Currently you have $N$ balloons, and you are trying to float the house into the sky by attaching balloons on the rooftop. Every balloon have altitude limit $L_{i}$ and capacity $D_{i}$, which indicates you can blow balloons in altitude at most $L_{i}$, and the balloon busts after increasing the altitude by $D_{i}$.
Your journey starts at altitude 0 . If you have more than 1 balloons enlarged, then the house will ascend too fast. Thus, you will blow one balloon and attach it at the rooftop, increase the altitude until the balloons bust, blow the other balloon and attach it to increase the altitude... to make your house float. For convenience, you may assume that balloons can only increase the altitude.

You don't care about your final altitude, but a balloon can move a fixed amount of distance. Thus, you
want to bust as many balloons as possible. You want to calculate a maximum number of balloons you can bust, and check if you can make a journey to KAIST. Let's see whether your 100-year-old ICPC experience can help on this problem!

## Input

The first line contains $N$, the number of balloons.
In next $N$ lines, the altitude limit of $i$-th balloon $L_{i}$, and capacity of $i$-th balloon $D_{i}$ are given as two space-separated integers.

## Output

Print the maximum number of balloons you can bust.

## Constraints

- $1 \leq N \leq 250,000$
- $0 \leq L_{i} \leq 10^{15}$
- $1 \leq D_{i} \leq 10^{9}$


## Subtask 1 (22 points)

This subtask has additional constraints.:

- $N \leq 20$


## Subtask 2 (33 points)

This subtask has additional constraints.:

- $N \leq 5,000$


## Subtask 3 (45 points)

This subtask has no additional constraints.

## Examples

|  | standard input |  | standard output |
| :--- | :--- | :--- | :--- |
| 3 |  | 2 |  |
| 1 | 4 |  |  |
| 1 | 5 |  |  |
| 9 | 2 | 1 |  |
| 4 |  |  |  |
| 0 | 1 |  |  |
| 0 | 2 | 3 |  |
| 0 | 4 |  |  |

## Problem U. United States Of Eurasia

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
20 seconds
1024 megabytes

In the year 5013, jh05013 conquered the Eurasian Continent and found "jh land". "jh land" encompasses the whole continent, and jh05013 aims to divide the country into "districts" to govern them efficiently, since it is extremely large and populated. There are $N$ houses in "jh land", and each house is located at $\left(x_{i}, y_{i}\right)$ of a 2D Cartesian coordinate system. jh05013 keeps the following conditions when dividing them into districts:

- Condition 1. Each district contains houses whose $x$ coordinate is in a certain range. (A district might contain all or none of the houses)
- Condition 2. Each house is contained in exactly one district.
- Condition 3. There are at most $K$ districts.

There are diverse races, religions, and nations in the Eurasian Continent. To avoid disputes, we must reduce the "splittedness" of districts. The splittedness of a district is defined as the distance between the farthest pair of houses contained in the district. The distance is measured in Euclidean metric. Help jh05013 minimize the maximum splittedness among districts.

## Input

In the first line, two space-separated integers $N, K$ are given. $N$ denotes the number of houses and $K$ denotes the number of districts.

In the next $N$ lines, two space-separated integers $x_{i}, y_{i}$ are given. They denote that a house is located at $\left(x_{i}, y_{i}\right)$.

## Output

When the minimum of maximum splittedness among districts is $M$, print $M^{2}$.

## Constraints

- $1 \leq K \leq N \leq 250,000$
- Every given location is distinct. It means, $i \neq j$ implies $x_{i} \neq x_{j}$ or $y_{i} \neq y_{j}$.
- $0 \leq x_{i}, y_{i} \leq 10^{9}$


## Subtask 1 (20 points)

This subtask has additional constraints.:

- $N \leq 500$


## Subtask 2 (20 points)

This subtask has additional constraints.:

- $N \leq 5,000$

Subtask 3 (60 points)
This subtask has no additional constraints.

## Examples

| standard input |  |  |
| :--- | :--- | :--- |
| 4 | 2 | 8 |
| 101 | 100 | standard output |
| 2 | 5 |  |
| 100101 | 3 |  |
| 4 | 3 |  |
| 4 | 4 | 0 |
| 3 | 1 |  |
| 4 | 1 |  |
| 5 | 1 |  |
| 9 | 2 |  |

## Problem V. Voronoi Diagram

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
1 second
1024 megabytes


Figure: Voronoi Diagram made with 20 points, with respect to Euclidean metric. Source: Wikipedia
In the Cartesian coordinate, we define the Voronoi Diagram of nonempty point set of size $n$ as a diagram that divides the plane by the criteria "which point in a set is closest in this location?". For example, in the picture above, every location over the plane is colored by the closest black point with such location. There is an algorithm that computes Voronoi Diagram in $\mathcal{O}(n \log (n))$, but this is infamous to be very complicated and hard.
After failing to solve a problem about Voronoi Diagram in an important competition, Minkyu was shocked, and he started living with alcohols everyday! One sunny afternoon, Minkyu was drinking beers just like the other day, and found an ingenious algorithm for solving Voronoi Diagrams! Before writing a paper about it, Minkyu wants to set a problem which requires this algorithm, to prevent any full scorers in 2018 KAIST RUN Spring Contest.
Why is Minkyu's algorithm for Voronoi Diagram great? Traditional algorithm for Voronoi Diagrams works only in cartesian coordinate, but Minkyu's algorithm works on more generalized structure - the "graph". Consider a connected graph with $N$ vertices and $M$ edges with positive weight. When you are given a nonempty vertex subset of size $K$, the "Voronoi Diagram" of this point set divides all location in the edges by the criteria "which vertex in a set is closest in this location?" If there is more than one points with equal distance, the one with smallest vertex number is considered closer.
You are given a weighted connnected graph and a nonempty vertex subset of size $K$. For each vertex, you should calculate the total length of edges that is "closest" to the given vertex. Solve this problem, and write the paper faster than Minkyu, to scatter his high hopes!

## Input

In the first line, the number of vertices $N$, and the number of edges $M$ are given as two space-separated integers.

In the next $M$ lines, two endpoint of the $i$-th edge $s_{i}, e_{i}$, and the weight of $i$-th edge $w_{i}$ is given as three space-separated integers.
In the next line, the size of vertex subset $K$ is given.
In the next line, $K$ distinct integer $a_{i}$ is given in increasing order. Each integer denotes the number of vertex in the subset.
You can assume that the given graph is connected. In other words, there exists a path from any vertex to the any other vertex.

## Output

In $K$ lines, print one decimal number for each line. In $i$-th line, print the sum of length which considers vertex $a_{i}$ as the closest.
Every outputted numbers should be exactly rounded to the first digit after the decimal point (see the sample input/output for clarification). In accordance to the recent ACM-ICPC World Finals trend (which requires high-precision floating point management), no precision error is allowed.

## Constraints

- $1 \leq N, M \leq 250,000$
- $1 \leq s_{i}, e_{i} \leq N$
- $1 \leq w_{i} \leq 10^{9}$
- $1 \leq K \leq N$
- $1 \leq a_{i} \leq N$
- Given graph is connected.


## Subtask 1 (27 points)

This subtask has additional constraints.:

- $N \leq 500$


## Subtask 2 (73 points)

This subtask has no additional constraints.

## Examples

|  | standard input |  |
| :--- | :--- | :--- |
| 3 | 3 |  |
| 1 | 2 | 5 |
| 2 | 3 | 5 |
| 3 | 1 | 5 |
| 2 |  | 7.5 |
| 1 | 2 | 7.5 |
| 5 | 4 |  |
| 1 | 2 | 10 |
| 2 | 4 | 20 |
| 3 | 4 | 30 |
| 4 | 5 | 50 |
| 2 |  | 80.0 |
| 1 | 3 |  |
| 11 | 10 | 30.0 |
| 1 | 2 | 1000000000 |
| 1 | 3 | 1000000000 |
| 1 | 4 | 1000000000 |
| 1 | 5 | 1000000000 |
| 1 | 6 | 1000000000 |
| 1 | 7 | 1000000000 |
| 1 | 8 | 1000000000 |
| 1 | 9 | 1000000000 |
| 1 | 10 | 1000000000 |
| 1 | 11 | 100000000 |
| 1 |  | 10000000000.0 |
| 1 |  |  |

## Notes

These are the pictures corresponding to each sample test.



Page 20 of 31

## Problem W. Winter Olympic Games

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
5 seconds
1024 megabytes


Figure: "Soohorang" - Not related to this problem, but included just because it's cute.
2018 RUN@KAIST Winter Curling Competition women's finals game is now ongoing. On the frozen "duck pond" of KAIST, Korean women curling team is having a fierce competition with team from country Jwepan!
There are $N$ curling stones on the "duck pond". As the competition is really fierce, every stone is placed in a line from a mark. The leftmost stone is closest from the mark, while the rightmost stone is farthest from the mark. Stones are either from Korean team (' 1 '), or from Jwepan team (' 0 '). Those arrangement of stones can be represented with length $N$ binary sequence.
After the end of Pyeongchang Olympics, Korean team had gone through intensive training. Now with some shoutings(?), team member "Youngmi", who carries the curling stone, can bounce away some consecutive stones and place her stone in that position. Formally, Korean team can pick any subsegment in a binary string (which can be empty), and replace it into a single digit " 1 ".
Korean team is a master in a curling strategy, and they knew the best strategy for one turn is to make the string lexicographically maximal! For the fast decision making in this game, they want to find a fastest algorithm which can find this. Help the Korean team to win the competition!
String $s=s_{1} s_{2} \cdots s_{n}$ of length $n$ is lexicographically larger than string $t=t_{1} t_{2} \cdots t_{m}$ of length $m$, if one of the following holds:

- There exists some $i$ such that, $s_{1}=t_{1}, s_{2}=t_{2}, \cdots, s_{i-1}=t_{i-1}$, and $s_{i}>t_{i}$.
- $n>m$ and, $s_{1}=t_{1}, s_{2}=t_{2}, \cdots, s_{m}=t_{m}$.


## Input

In the first line, $N$, the number of stones is given.

In the second line, A single binary string of length $N$, which consists of ' 0 ' or ' 1 ' is given. This string indicates the owner of each curling stone, in the order of distance from the mark. There are no quotes or blanks given in the string.

## Output

Print two integer $S, L$. This means that Youngmi removed $L$ stones after $S$ th character. If there is more than one correct answer, print any. $(0 \leq S, L \leq N)$

## Constraints

- $1 \leq N \leq 1,000,000$


## Subtask 1 (23 points)

This subtask has additional constraints.:

- $N \leq 500$


## Subtask 2 (24 points)

This subtask has additional constraints.:

- $N \leq 10,000$


## Subtask 3 (53 points)

This subtask has no additional constraints.

## Examples

| standard input | standard output |  |
| :--- | :--- | :--- |
| 8 <br> 10101101 | 13 |  |
| 5 | 0 | 0 |

## Problem X. Xtreme NP-hard Problem?!

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
5 seconds
1024 megabytes

Caution! This problem turned out to be NP-hard. But since there was no rules against writing a NP-hard problem, we decided to leave this problem here.
There is a bidirectional graph consisting of $n$ vertices and $m$ edges. The vertices and edges are numbered from 1 to $n$ and 1 to $m$ respectively, and the weight of edge $i$ is $w_{i} .(1 \leq i \leq m)$ Given a natural number $k$, find the length of the shortest simple path that starts from vertex 1 and ends at vertex $n$, and consists of $k$ edges. A simple path is a path that does not visit same vertex twice, and length of a path is the sum of weight of edges that consists the path.

## Input

In the first line, three space-separated integers $n, m, k$ are given.
In the next $m$ lines, three space-separated integers $x_{i}, y_{i}, w_{i}$ are given. They denote that edge $i$ is connecting vertex $x_{i}$ and vertex $y_{i}$, and has weight $w_{i}$.
No loops or multiple edges are given.

## Output

Print the length of the shortest simple path that starts from vertex 1 and ends at vertex $n$, and consists of $k$ edges. If there is no such path, print -1 .

## Constraints

- $2 \leq n<10^{6}$
- $1 \leq m, k<10^{6}$
- $1 \leq x_{i}, y_{i} \leq n$
- $x_{i} \neq y_{i}(1 \leq i \leq n)$
- $i \neq j \Longrightarrow\left\{x_{i}, y_{i}\right\} \neq\left\{x_{j}, y_{j}\right\}(1 \leq i, j \leq n)$
- $1 \leq w_{i} \leq 10^{8}$


## Subtask 1 (19 points)

This subtask has additional constraints.:

- $\min (n, m, k) \leq 3$


## Subtask 2 (27 points)

This subtask has additional constraints.:

- $\min (n, m, k) \leq 4$


## Subtask 3 (54 points)

This subtask has additional constraints.:

- $\min (n, m, k) \leq 5$


## Examples

|  |  | standard input |  | standard output |
| :--- | :--- | :--- | :--- | :--- |
| 6 | 6 | 3 | 8 |  |
| 1 | 2 | 3 |  |  |
| 2 | 3 | 1 |  |  |
| 3 | 6 | 4 |  |  |
| 1 | 4 | 1 |  |  |
| 4 | 5 | 5 |  |  |
| 5 | 6 | 9 |  |  |

## Problem Y. Yut Nori

Input file:
Output file:
Time limit:
Memory limit:
standard input
standard output
1 second
1024 megabytes

Yut Nori (Yut Game) is one of famous played with Yut. Yut Nori is played with 4 yuts, one Yut board, and 8 tokens, 4 for each two team. Yut board has 29 stations. Each station has its own name. (Figure 1)


Figure 1: Yut board. There is name for each stations.


Figure 2: possible four routes.
Goal of the game is starting from cham-meoki, walk through the board and return to cham-meoki and escape.
Turn of each person is as follows:

- Throw 4 Yut.
- Determine the number of steps based on Yut.
- Select the token and walk through the board.

Yut has front-side and back-side. If Yut is thrown, its front-side or back-side is shown. The number of steps is the number of front side. However, if every side is back-side, the number of steps is 5 .
The new token starts at cham-moeki (it is not placed actually, but assume the token is placed so), and follows these Yut-gil(route).
Basically, tokens move through the fourth route (Figure 2). Four example, if you walk two steps from yut, it goes to duet-do. But in special cases, it goes through first, second, or third route. If it starts from corner (mo, duet-mo, bang, chi-mo, or cham-meoki), it selects faster way to cham-meoki.

More formally, if token starts moving at mo, route changes to third route. For example, if you walk one step from mo, it goes to mo-do, two steps than mo-gae. If you walk one step twice from yut, it goes to mo-do, but if you walk two steps from yut, it goes to duet-do.

Also, if you start from duet-mo, token follows the second route, so walking one step leads to duet-modo, while walking two step leads to duet-mogae.
Moreover, if token following the third route starts from bang, then token follows the first route. Walking one step goes to saryeo, two step goes to anchi.
Tokens moving like these escapes the board after it arrived cham-meoki. (It does not requires separate steps) For example, if you walk two steps from nal-geol, it arrives to cham-meoki, and walking one step more makes the token escape.
If you walk three or more steps from nal-geol at once, it also escapes the board.
Until now, this is the game that just throws Yut to escape. However, there are two more rules to make the game more fun.
First rule is, move together. If there are two or more tokens at the same station, moving one of them will move every token.
Second rule is, token catch. If opponents' token is placed at the destination of your token, move all of opponents' token to initial state.
You are given a task for development of Yut Nori. You have to replay the game.

## Input

The first line of input consists $N$. This is the number of the turn.
In $i$ th line of following $N$ lines, the information of $i$ th turn is given. The information of $i$ th line is given as information of token and yut.
The information of token is given as one character, one of "ABCDabcd". The information of Yut is given as a string with four characters, each character is ' $B$ ' for back, and ' $F$ ' for front. The information of token and Yut is space-separated.
The tokened are teamed with upper-case and lower-case.

## Output

Basic Yut board is given as following $32 \times 32$ string.:


This string consists of " $\backslash$. $\mid$ ", space and newline character. (quote is not included.) This string denotes Yut board, each station is represented as $2 \times 2$ '.' character. From upper-leftmost character, stations are duet-mo, duet-yut, duet-geol, duet-gae, duet-do, mo, duet-modo, mo-do, chi-do, yut- duet-mogae, mo-gae, chi-gae, geol, bang, chi-geol, gae, sok-yut, saryeo, chi-yut, do, sok-mo, anchi, chi-mo, nal-do, nal-gae, nal-geol, nalyut, and cham-meoki, read row major order.
You should print whether the token is in this board. You should replace one of '.' character for the token in station.
A or a should replace upper-left, B or b should replace upper-right, C or c should replace lower-left, and D or d should replace lower-right '.' of the board.

## Constraints

- $1 \leq N \leq 100$
- Every move is valid; escaped token will not be given again at board.


## Subtask 1 (31 points)

This subtask has additional constraints.:

- $N=1$
- Token will not be placed at any corner during the game.
- Every token in input will be one of "ABCD" chracters. (Without quotes.)


## Subtask 2 (13 points)

This subtask has additional constraints.:

- There will be no two tokens at the same stations during the game.
- Token will not be placed at any corner during the game.
- Every token in input will be one of "ABCD" chracters. (Without quotes.)


## Subtask 3 (10 points)

This subtask has additional constraints.:

- Token will not be placed at any corner during the game.
- Every token in input will be one of "ABCD" chracters. (Without quotes.)


## Subtask 4 (27 points)

This subtask has additional constraints.:

- Every token in input will be one of "ABCD" chracters. (Without quotes.)


## Subtask 5 (19 points)

This subtask has no additional constraints.

## Examples



| standard input | standard output |
| :---: | :---: |
| 21 | ..----..----..----...----..----.B |
| A FFFB | .. cd C. |
| c FBBF | । $\$ /  \hline d FBBB & 1 \ / l  \hline B BBFB & । \ / \|  \hline C BFBF & \| ab  \hline d FFBF & .. .. .. A.  \hline B BFBF & .. $\$ / ..  \hline a BBBF & 11  \hline C BBBB & 1 /  \hline D FBFF & I .. .. \|  \hline a FFFB & I .. .. \|  \hline d FBFF & .. 1  \hline b FFFF & ..  \hline b BBBB & \/  \hline c BFFF & I .. \|  \hline B BBFB & I .. \|  \hline b BBBF & $1 / 1$ |
| B FFFF | .. / \} |
| c BBBB | .. / \} |
| b BBBF | I .. .. \| |
| A FFFF | 1 |
|  | 111 |
|  | 111 |
|  | $\cdots 1$ |
|  | I .. .. \| |
|  | 11 |
|  | 11 \ 1 |
|  | $1 /$ \ 1 |

## Problem Z. Zigzag

Input file:
Output file:
Time limit:
Memory limit
standard input
standard output
1 second
1024 megabytes

A sequence is called "Zigzag" if no three of its consecutive elements are monotone.
More formally, if sequence $A$ of length $N$ is Zigzag if, for all $i(1 \leq i \leq N-2)$, neither $A_{i} \leq A_{i+1} \leq A_{i+2}$ nor $A_{i} \geq A_{i+1} \geq A_{i+2}$ holds.
For given sequence $A$ of length $N$, you should find a longest subsegment of $A$ which is a Zigzag sequence.
Sequence $B$ of length $M$ is subsegment of sequence $A$ of length $N$ if, for some $i, B_{1}=A_{i}, B_{2}=A_{i+1} \cdots$, $B_{M}=A_{i+M-1}$ holds.

## Input

Input consists of two lines.
The first line contains integer $N$, length of sequence $A$.
The second line contains space-separated $N$ integers. $i$ th number is $A_{i}$.

## Output

Print out the length of longest subsegment of $A$ which is a Zigzag sequence.

## Constraints

- $3 \leq N \leq 5,000$
- $1 \leq A_{i} \leq 10^{9}(1 \leq i \leq N)$


## Subtask 1 (33 points)

This subtask has additional constraints.:

- $N=3$


## Subtask 2 (37 points)

This subtask has additional constraints.:

- $N \leq 300$


## Subtask 3 (30 points)

This subtask has no additional constraints.

## Examples

|  | standard input | standard output |  |  |
| :--- | :--- | :--- | :--- | :--- |
| 3 |  |  |  | 2 |
|  |  |  |  |  |
| 1 | 3 | 4 | 2 | 5 |

