Irish Collegiate Programming Competition 2023

Problem Set

ACM Student Chapter, University College Cork

March 25, 2023

Instructions

Rules

- All laptops, and other electronic devices must be powered off and stowed away for the duration of the contest. Mobile phones should also be stowed in the provided ziplock bags and left on your desk for the duration of the competition. Use of custom hardware (keyboard, mouse, ...) is not permitted.
- Reference books, personal notes or sheets are not permitted. You will be provided with the documentation of the supported languages.
- The only networked resource that teams are permitted to access is the submission system.
- No multi-threading is allowed, and no sub-processes.
- Use of non built-in libraries and modules, such as Python NumPY, are not supported by the server.
- No file input/output is allowed. All input to your program will be provided via standard input (stdin) and all output should be printed to standard output (stdout). Examples of how to do this in each of the languages is provided in the resources section of the submission site.
- If a team discovers an ambiguity or error in a problem statement, they should submit a question in the submission system. If the organisers agree that an ambiguity or error exists, a clarification will be issued to all teams.
- Toilet breaks: You will be allowed to leave the exam room to go to the toilet (but please go beforehand). Raise your hand and wait for an organiser to come to you.

Submission Instructions

- Your password will be provided by the organisers. Notify an organiser if you are unable to log in.
- Submissions should consist of a single source file, **not a compiled executable**.
- To submit, click the "Submit a Solution" link, complete the submission form, upload your source file, and click "Save". Your submission should now be listed in your submission queue.
- Java solutions should be a single source file and should not include the package header. The main class name should match the name of the task; for example, for the task "adding_integers", the main class should be defined as: `public class adding_integers {...}`
- C (C11) and C++ (C++11) submissions will be compiled with the flags `-lm -O2`.

Testing and Scoring

- Solutions should be submitted in the form of source code, which will be compiled on the test environment.
- The output from your program should be terminated by a new line and should not contain any additional whitespace at the beginning or end of lines.
- Except if specified otherwise in a problem's text, each solution will be tested against 20 separate test cases. Some are designed to test the limits and corner cases of the input. Test cases are not necessarily ordered in any way. Five points is given for each correct output.
- Programs are limited to the CPU time and RAM specified within the question's text. These limits hold for each test case (compilation is not included in those restrictions).
- If a solution is submitted while an earlier solution to the same problem is in the queue, the earlier submission will not be tested and will be marked with the message: "Old, not going to be tested".
- A teams total score is based on the highest scoring submission for each question, regardless of lower scoring subsequent submissions.
- Tiebreaker. The winning team will be the one whose last scoring submission was submitted earliest.

Contributors

The UCC ACM student chapter would like to warmly thank the following people for helping with the question writing:

- Guillaume Escamocher
- Andrew Nash
- Bastien Pietropaoli
- Andrea Visentin

1 Arthur's Audacious Alien Abduction (CPU:1sec - RAM:256MB)

Arthur (ze/hir), an alien scientist from the Cadence planet, just discovered a small rock planet filled with incredible creatures. One creature in particular tickles hir curiosity. They are present all over the planet and strongly modify the areas in which they live. For Arthur, they all look extremely alike, with minor variations in size, skin and fur colour. Ze can't wait to collect some specimens and return to hir home planet to study them.

The abduction laser is ready, and Arthur has prepared a few alternative plans to collect these creatures. Each plan divides the world into N segments; each segment contains a number of specimens to be collected. The abduction laser spans K consecutive segments. Every time it is fired, it can collect only one creature from each segment spanned by the laser.

To choose the best plan, Arthur needs to know the minimum number of times the abduction laser must be fired to collect every specimen. Can you help hir?

Input The input consists of the following lines:

- A line consisting of a single integer N, with $0 \leq N \leq 700000$, which indicates the number of segments in the plan.
- A line consisting of a single integer K, with $0 \leq K \leq 125000$, which indicates the number of consecutive segments that the laser can span.
- Followed by N lines, each describing a segment. Each of these N lines contains an integer i (where $0 \leq i \leq 125000$) that indicates the number of specimens contained in that segment.

Output The output should consist of a single line containing an integer equal to the minimum amount of times the laser has to be fired to collect every specimen.

Special case In 30% of the plans, it is possible to always fire so that the laser spans over segments with at least one specimen in each.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

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

Sample Output 1

```
5
```

2 Caitríona's Calendars (CPU:1sec - RAM:256MB)

Caitríona likes to take her vacations in exotic parts of the Expansive InterstellaR Empire (EIRE). To avoid crowds, she always uses a calendar to plan her trips around weekends and local bank holidays. However, since each planet of the EIRE has its own culture and revolution (movement around its star) period, the calendars are all different!

Caitríona has heard good things about your vacation planning skills from her friend Stella, so she decides to ask you for help. For a given calendar and date, you have to determine which day of the week that particular date falls on. This would have been easy enough, if not for the fact that many planets have adopted a leap year system to improve the accuracy of their calendars. Fortunately, all such planets use the same template for leap years: a day is added at the end of the second month of each leap year, and a year is a leap year if and only if it is divisible by X or (it is divisible by Y but not by Z). Only the values of X, Y and Z vary between planets (for example on Earth the values are $X = 400$, $Y = 4$ and $Z = 100$).

Input The input consists of the following lines:

- A line consisting of five integers separated by a single space that describe a calendar: the number $2 \leq W \leq 10$ of days in a week, the number $2 \leq M \leq 20$ of months in a year and the three parameters $1 \leq X \leq 1000000$, $1 \leq Y \leq 10$ and $2 \leq Z \leq 100$ describing the leap year system.
- Followed by a line containing M integers $m_1, m_2, \ldots, m_M$ separated by a single space. For each i, m_i is the number of days in the i^{th} month (on a non leap year) and $2 \leq m_i \leq 100$.
- Followed by a line containing a single integer indicating the number $1 \leq N \leq 100$ of dates that will be queried.
- Followed by N lines each containing a date composed of three integers a, b and c separated by a single space with $1 \leq a \leq 10^{12}$ being the year, $1 \leq b \leq M$ being the month and $1 \leq c$ being the day. You can assume that c is a valid day for that particular month, that is $c \leq m_b$ (or $c \leq m_b + 1$ if a is a leap year and $b = 2$). Furthermore you can also assume that all N dates are different and given in chronological order.

Output The output shall consist of N lines, each consisting of a single integer indicating the day of the week of the corresponding date.

Conveniently the first day of the first month of the first year in all calendars is a Monday (or the cultural equivalent). In other words, the output for the date input 1 1 1 is always 1.

Sample Input 1

```
7 12 400 4 100
31 28 31 30 31 30 31 31 30 31 30 31
3
1776 7 4
1958 1 1
2022 3 25
```

Sample Output 1

```
4
3
6
```

Sample Input 2

```
8 5 500 6 50
2 19 5 19 2
2
1 1 1
1000 2 20
```

Sample Output 2

```
1
8
```

3 ChatGPT Crawl (CPU:1sec - RAM:256MB)

This question has been generated with ChatGPT with minimum edits.

You are organizing a pub crawl in Dublin, and you want to visit as many pubs as possible while spending the least amount of money. You have a list of N pubs to visit, each with a cost of c_i euro per visit. You have a budget of B euro for the pub crawl.

You can only visit each pub once, and you must visit them in order from the first to the last pub on the list. However, you can skip any number of pubs along the way (including the first and last pubs) if you decide that you don't want to pay their charges.

Write a program that takes a positive integer N representing the number of pubs, a list c representing the cover charges for each pub, and a positive integer B representing your budget. The program should print the maximum amount of pubs you can visit.

Input The input consists of the following lines:

- A line consisting of a single integer N, with $0 \leq N \leq 500000$, which indicates the number of pubs.
- A line consisting of a single integer B, with $0 \leq B \leq 5000000$, which indicates the budget.
- Followed by N lines, each containing c_i the cost of pub i, with $0 \leq c_i \leq 50$.

Output The output should consist of a single line containing an integer representing the maximum number of pubs that can be visited.

Sample Input 1

```
3
10
2
6
10
```

Sample Output 1

```
2
```

Sample Input 2

```
8
40
18
2
8
10
16
7
19
15
```

Sample Output 2

```
4
```

4 Donagh Daunting Deliveries (CPU:1sec - RAM:50MB)

Donagh has opened a new service: DeliverZoo. He delivers animal food, medicines and toys to animal owners in Cork. Business is going excellently, and he has many requests. His main problem is that he has only one truck to do all the deliveries. Each day, he drives the truck from the warehouse to the supplier to pick up a daily supply of goods and returns to the warehouse. He wants to make some cleverly chosen stops along the way to deliver as many products as possible to his list of customers. For maximum productivity, he makes deliveries on both legs of the journey - both from the warehouse to the supplier, and from the supplier to the warehouse.

Your task is to prepare Donagh's itinerary for the day. The simplified layout of Cork is represented by a grid of $(N \times N)$ squares. The warehouse is located in $(0, 0)$ (top left) and the supplier in $(N-1, N-1)$ (bottom right). Due to a new traffic management scheme, when driving to the supplier Donagh can go only down or right on the grid (no moves going upwards or leftwards on the grid are allowed). After reaching the supplier, Donagh can make the return journey by only going up and left. He can stop to deliver any products he wants during either journey, so long as the above rules are not violated. Three numbers represent each customer: x, y, z, where x is the index of the row of the customer's house, y is the column index, and z is the order value in Euro. Each cell can contain at most one order, and each order can be satisfied only once.

Can you tell Donagh the maximum amount of money he can make during his ride?

Input The input consists of the following lines:

- A line consisting of a single integer N, with $0 \leq N \leq 250$, which indicates the size the grid.
- A line consisting of a single integer O, with $0 \leq O \leq 5000$, which indicates the number of orders.
- Followed by O lines, each describing a particular order. Each of these O lines contains three integers (x, y, and z separated by a space. x and y indicate the row and column where the customer is located, and z indicates the order value, with $0 \leq z \leq 50$.

Output The output should consist of a single line containing an integer representing the maximum value of the orders that can be delivered during the path.

Sample Input 1

```
3
3
2 1 9
1 2 7
0 2 6
```

Sample Output 1

```
22
```

Sample Input 2

```
3
7
1 0 2
1 1 5
2 0 1
1 2 5
0 1 4
2 1 5
0 2 3
```

Sample Output 2

```
24
```

5 Fiona's Fun Fairies (CPU:1sec - RAM:256MB)

Fiona is an avid chess player. Not boring plain chess, but the much more exciting fairy version. In this variant, the pieces are not the well-known pawn, knight, bishop, rook, queen and king, but instead fairy pieces with different movement rules. The same fairy piece can be referred to by more than one name, if it has multiple origins from different cultural backgrounds. For example, one popular fairy piece is a combination of a queen and a knight, and has been called Amazon, Empress, Maharajah, as well as many other terms.

A chessboard in fairy chess is the same as in standard chess: a grid of 64 squares with eight columns labelled A to H from left to right, and eight rows labelled 1 to 8 from bottom to top. Fairy pieces can never move to a square containing a piece with the same color but, unless specified otherwise below, they can move to an empty square, or to a square containing a piece of the opposite color, which is then said to be captured.

Here are the names Fiona gave to the pieces she most often plays with:

- The following pieces are *hoppers*. They move orthogonally (either within the same column or within the same row, like a rook), and there must be exactly one piece, of either color, between them and their destination.
  - **frog**: the intermediate piece must be exactly one square away from the starting square. For example, if the frog is in (C,7) and the only other piece on the chessboard is in (C,6) then the squares that the frog can move to are the five ones from (C,5) to (C,1).
  - **rabbit**: the intermediate piece must be exactly one square away from the destination square. For example, if the rabbit is in (C,7) and the only other piece on the chessboard is in (C,4) then the only square that the rabbit can move to is (C,3).
  - **kangaroo**: the destination piece must be at the exact same distance from both the starting square and the destination square. For example, if the kangaroo is in (C,7) and the only other pieces on the chessboard are in (C,4) and (D,7) then the only squares that the kangaroo can move to are (C,1) and (E,7).
- **ram**: moves orthogonally, and only towards the top and/or right. Also, no other piece should be present between the starting and destination squares. For example, if the ram is in (C,7) and the only other piece on the chessboard is in (G,7) and of the same color as the ram, then the squares that the ram can move to are (C,8), (D,7), (E,7), and (F,7).
- **camel**: can move to any square that is either exactly 1 column and 3 rows away, or exactly 1 row and 3 columns away. The camel can leap over any intermediate piece. For example, if the camel is in (C,7), then the squares that the camel can move to are (B,4), (D,4), (F,6) and (F,8), regardless of whether there are other pieces in column E or row 5.
- **zebra**: same as the camel, but with 2 and 3 away instead of 1 and 3. For example, if the zebra is in (C,7), then the squares that the zebra can move to are (A,4), (E,4), and (F,5).
- **nightrider**: can make any number of (1,2) leaps, as long as they all are in the same direction. There cannot be any other piece on the intermediate squares leaped *to*, but there can be pieces on the squares leaped *over*. For example, if the nightrider is in (C,7) and the only other pieces on the chessboard are in (B,6), (C,6), and (E,3), all three of the same color as the nightrider, then the squares that the nightrider can move to are (A,6), (B,5), (A,3), (D,5), (E,6), (G,5), and (E,8).
- **crow**: can swap positions with any piece of the other color, as long as the target piece is not within distance 1, including diagonally, of another piece of its color. For example, if the only pieces other than the crow are in (C,2), (D,3) and (D,5), and none of them is of the same color as the crow, then the only square that the crow can move to is (D,5).
- **leopard**: can go orthogonally left, right, down or up any number of times, and it does not have to always be in the same direction, but the leopard can never be within distance 1, including diagonally, of any other piece, unless it is a piece of the other color that is present on the destination square and is captured by the leopard. Under no circumstances can leopards start moving if they are already within distance 1 of another piece.

- **bat**: moves orthogonally, but can move only horizontally if capturing and only vertically if not. The bat cannot leap over intermediate pieces. For example, if the bat is in (C,7) and the only other pieces on the chessboard are in (C,2) and (H,7), and none of them is of the same color as the bat, then the only squares that the bat can move to are (C,3), (C,4), (C,5), and (C,6), which are empty squares in a vertical direction, and (H,7), which is a horizontal capture.

There is a leprechaun fairy piece, but it is so powerful that any game with it is quickly decided, so Fiona prefers not to use it.

Given the positions of the pieces on a chessboard, and one particular fairy piece on this same chessboard, Fiona wants to find all squares that this fairy piece can move to.

Input The input consists of the following lines:

- Eight lines, one for each row of the chessboard, from the top row 8 to the bottom row 1. Each line consists of eight space separated characters, one for each square of the row. Each character can only be 'W', 'B', or '0'. 'W' indicates that a white piece is present on the square, 'B' indicates that a black piece is present on the square, and '0' indicates that the square is empty.
- One line consisting of two characters, the first one is a capital letter (A-H) corresponding to the column of the fairy piece to move, and the second one is a digit (1-8) corresponding to the row of the fairy piece. You can assume that the character from the previous lines that corresponds to this square is either 'W' or 'B'.
- One line consisting of the name of the fairy piece being analysed. This line only contains lower case letters (a-z) and can only be the name of one of the ten pieces used by Fiona: frog, rabbit, kangaroo, ram, camel, zebra, nightrider, crow, leopard or bat.

Output Eight lines, one for each row of the chessboard, from row 8 to row 1. Each line consists of eight space separated digits, one for each square of the row. Each digit can either be 1 or 0, the former to indicate that the highlighted fairy piece can move to this square, and the latter to indicate that it cannot.

The square on which the fairy piece originally is should be marked with 0 if there is at least one possible move to a different square, and with 1 if on the other hand the fairy piece has no choice but to stay in place.

Sample Input 1

```
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 B
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 B W W W 0
0 0 0 0 W W W 0
0 0 0 W W W W W
0 0 0 0 0 0 B 0
F3
nightrider
```

Sample Input 2

```
0 0 0 0 0 0 0 0
W 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
B B B B B 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 0 0 0 B
A7
leopard
```

Sample Input 3

```
0 B 0 0 0 0 0 0
W 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
B B B B B 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 W 0 0 0
0 0 0 0 0 0 0 B
A7
leopard
```

Sample Input 4

```
B B B 0 0 0 0 B
B W B 0 0 0 0 0
B B B 0 B 0 B 0
0 0 0 0 0 0 0 0
W W W W W W W W
0 0 0 0 0 0 0 0
0 0 W 0 0 0 0 0
0 0 0 0 0 0 0 0
H8
crow
```

Sample Output 1

```
0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0
0 0 0 1 0 0 0 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0
```

Sample Output 2

```
1 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1
```

Sample Output 3

```
0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

Sample Output 4

```
0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0
```

6 Gráinne's Gomoku Games (CPU:1sec - RAM:256MB)

Gráinne's new favourite pastime is the game of Gomoku. In this Japanese board game, two players place stones on a grid until one player manages to create a winning line: five consecutive stones of the same colour (either black or white) in a straight line. For added fun, Gráinne and her friends usually eschew Go boards, on which Gomoku is traditionally played, and use an infinite[1] grid instead. While this makes Gomoku games cooler, it also means it can be hard to keep track of the current state of the board, and in particular of whether there exists a winning move for the current player.

Your task is to save Gráinne time by quickly determining whether a given board contains an opportunity for the current player to complete a straight line of sufficient length and win the game, and if so to indicate where it is. Winning lines can be horizontal (all five stones in the line are in the same row), vertical (same column) or diagonal (two possible directions: top left to bottom right and bottom left to top right). Gráinne assures you that she will only give you boards from valid games. This means in particular that no two stones share the same coordinates and that either there are exactly as many black stones as there are white ones or there is exactly one more black stone (the player with black stones is the first to move). It also means that the potential winner (if there is one) on the current turn was not the one to place a stone on the previous turn. In warm-up games, Gráinne and her friends often try uncommon and exotic moves, so you have no guarantee on the actual soundness of the strategies employed.

Input The input consists of the following lines:

- A line consisting of a single integer N, with $0 \leq N \leq 100000$, which indicates the number of stones on the board.
- Followed by N lines, each describing a particular stone. Each of these N lines contains a letter (either B for a black stone or W for a white stone) followed by a space followed by an integer i (where $-1000000 \leq i \leq 1000000$ indicates in which column the stone is) followed by a space followed by an integer j (where $-1000000 \leq j \leq 1000000$ indicates in which row the stone is).

The stones are given in the order that they were played, which means that their colours alternate and that the first stone given in the input is black, while the last stone listed is white if the number of stones is even, and black if the number of stones is odd.

Output If there exists a cell that would complete a winning line if the current player places a stone there, you shall output the coordinates i and j of the cell separated by a single space, with i being the column of the cell and j being its row.

If there is no winning move for the current player on this turn, you only need to output the single integer 0.

You can assume that the game has not been won yet, or in other words that there is no straight line of five stones of the same color on the board. You can also assume that there is at most one winning cell for the current player, although placing a stone in this cell could complete more than one winning lines.

There could exist one or more cells that would complete a winning line for the other player. These should be ignored, as we are only interested in winning cells for the player who is currently making a move.

[1]In this context, "infinite" means "very large". Specifically, Gráinne uses a 2000000×2000000 grid

Sample Input 1

```
10
B 4 -7
W -2 -1
B -1 -2
W -3 -1
B 2 -5
W -2 1
B 0 -3
W -1 -1
B 1 -4
W -2 0
```

Sample Output 1

```
3 -6
```

Sample Input 2

```
13
B 1 4
W -2 4
B 1 3
W -1 3
B 1 2
W 0 2
B 1 1
W 2 0
B 2 1
W 3 -1
B 3 1
W 4 -2
B 4 1
```

Sample Output 2

```
0
```

7 Liam's Letter Learning (CPU:1sec - RAM:256MB)

Liam loves languages. He spends most of his free time studying foreign dialects, and is amazed at how each are unique in their own ways. Liam has noticed in particular that the relative frequencies of different letters vary from one language to another, with some languages seeming to favor certain letters that are rarely seen in others.

Liam now wants to classify languages according to their letter distribution. To do so, he will pick a culturally significant text for each language and count the number of times each letter appears. Liam is not interested in capitalisation rules, so he will treat both lower and upper case letters as if they were the same.

Input The input consists of the following two lines:

- A line consisting of a single integer N, with $1 \leq N \leq 20000000$, which indicates the length of the sample text.
- Followed by a line containing N characters. Each character is either a lower case letter (a-z) or an upper case one (A-Z). No digit or special character will be present in this line.

Output The two letters with the most occurrences in decreasing order of frequency. The letters must be given in upper case, even if they appear more frequently in lower case than in upper case in the sample text, and even if they do not appear in upper case at all.

The number of occurrences of a given letter is defined by the combined numbers of lower and upper case of this letter. So for example, a letter that appears three times as lower case and twice as upper case is considered to be more frequent than a letter that appears four times as upper case.

You can assume that there are no two letters tied for the most occurrences, and similarly that there is only one letter with the second most appearances.

Sample Input 1

```
15
aaaBBBBAAAccccc
```

Sample Output 1

```
AC
```

Sample Input 2

```
13
Cadenceishere
```

Sample Output 2

```
EC
```

8 Wendy's Western Windmills Whereabouts (CPU:1sec - RAM:256MB)

Wind power generation is a mature technology that consists in transforming wind's kinetic energy into electricity via the use of wind turbines. Ireland, with more than 300 wind farms, has one of the highest electric grid penetration values in the world. In order to stay on top of it (the world), Wendy has decided to continue exploiting the strong Irish winds by installing new farms on the West coast of Ireland.

In order to get the most out of the new wind farms, numerous locations are considered. For each of them, L windmills should be placed and it will be up to you to provide the best suitable locations for all of them to be installed. To help make your decision, Wendy provides you with a 2D map of each location in the form of a grid of $N \times M$ cells. $0 \leq n \leq N-1$ corresponds to the latitude at which a windmill may be placed, and $0 \leq m \leq M-1$ corresponds to the distance from the shore. The wind is considered to blow from the ocean (left) into the land (right).

The difficulty resides in multiple factors. Firstly, the wind is not constant over the grid and will depend on the latitude. The velocity of the wind at a given location is given by the following equation:

$$V_{wind} = a + bn + cn^2 - dm \tag{1}$$

in which n corresponds to the latitude (row) considered, m corresponds to the depth into the land (column), and a, b, c, and d are four constants that will be provided with each map. The power produced by a wind turbine is proportional to the cube of the speed of the wind it receives ($E \propto C \times V^3_{wind}$, you can assume C to be 1).

The second difficulty resides in that each turbine placed will potentially affect others. Since part of the wind's energy is absorbed by the windmill to generate power, when a turbine is placed, it will reduce the wind speed for every subsequent cell in the same line by 20%. For instance, if a first windmill is placed on the first cell of a line, and then a second one is placed on the third cell, then the second cell receives only 80% of its nominal wind, and the fourth and onwards will receive 64% (0.8 x 0.8) of their nominal wind.

Finally, turbines should probably not be placed next to one another. Two turbines placed in two neighbouring cells (same line or same column, diagonals are left unaffected) will see their power production reduced by 5% (again, multiplicatively) each due to local perturbations created by the turbines.

The objective for you here is thus to provide an optimal deployment plan in terms of power output for each map provided. Obviously, two windmills cannot be placed at the same location.

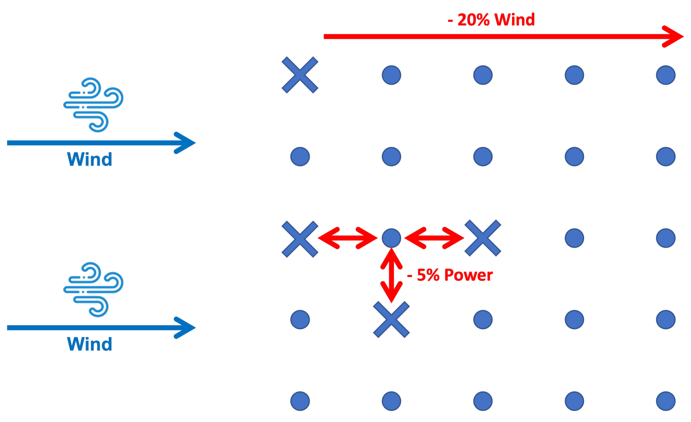

Figure 1: Here is the diagram provided to you by the engineer in charge, to help you understand the problem. Let's hope it helps.

This question is an optimisation problem. Since our solution is probably not optimal (though it might be), your solution will be graded compared to ours using the following rules:

- **If your solution provides power between 0 and ours, you will score up to 8 points proportionally of the quality of your solution with increasing returns.**
- **If your solution is better than ours, you will score from 8 to 10 points proportionnaly of the quality of your solution with diminishing returns.**
- **Your solution will be tested against 10 test cases only (each providing up to 10 points).**

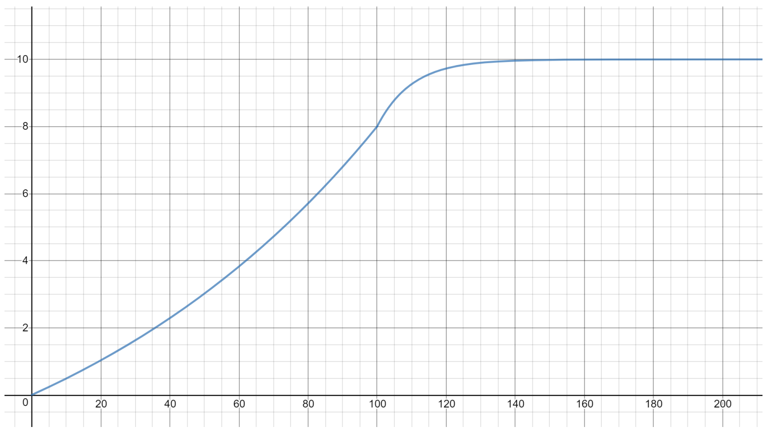

Figure 2: Graph of the grading function for this problem. The x-axis represents the proportion of your solution output relative to ours in percents. (e.g. If your solution provides a plan 50% as good as ours on a test case, it will score 3 points. It will provide 8 points if it's as good as our solution, and 9.5 points if it's 20% better than ours.)

Input

The input will consist of a single line containing the following space-separated numbers:

- N and M, two integers indicating the number of rows and columns (respectively) in the map, $1 \leq N, M \leq 500$.
- L, an integer indicating how many turbines should be placed, $L \leq N \times M$.
- a, b, c, and d, four integers indicating the parameters used to compute the velocity of the wind depending on latitude and depth into the land, $-1000 \leq a, b, c \leq 1000$, $0 < d \leq 1000$. At least one of a, b, and c will be a non-zero value.

Output

The output should be the optimal wind turbine deployment plan in terms of power generated in the form of N lines, each of them containing M characters. Each character will be a dot (`.`) if the cell is left empty, an `x` if a turbine should be placed into that cell. The N lines should be ordered by decreasing latitude.

Sample Input 1

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

Sample Output 1

```
xxx
...
...
```

Sample Input 2

```
5 10 8 8 4 1 1
```

Sample Output 2

```
x.x.x.x...
.x.x.x....
x.........
..........
..........
```