Problem A. Space Power

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Space Team needs to get the ship from point A to point B as fast as possible! Not only does this require them to calculate the shortest path through spacetime from A to B, but it also requires them to have as much engine power as possible!

The ship is powered by N thrusters. Each thruster has a different amount of power it can output. The power of the i-th thruster is given by V_i.

Each thruster can be put in three states: full forward, completely off and full backwards. Initially, thruster i is in state P_i where $P_i \in \{-1, 0, 1\}$ and -1 indicates reverse, 0 indicates off and 1 indicates full forward.

The Team has the controls to each of the thrusters and can change their state to maximise forward thrust. Ideally the Team would like to simply switch all the thrusters to full forward. However, past experience shows that the computer system has a serious bug. If they we don't perform exactly Q *state changes*, the system will freeze up. A state change consists of changing a thruster from full reverse to off (or vice versa) or from full forward to off (or vice versa).

Moreover, since the Team has been in space for quite a while now, each thruster only has a limited number of state changes left before it overheats. The i-th thruster may have at most S_i state changes applied to it in this procedure.

Help the Team determine the maximum power they can engage!

Input

The first line contains two integers, N and Q.

The second line contains N integers. The i-th number is V_i.

The third line contains N integers. The i-th number is P_i.

The last line contains N integers. The i-th number is S_i. It is guaranteed that the sum of the S_i is at least Q.

Output

Output a single integer, the maximum forward thrust possible.

Constraints

$1 \leq N \leq 200\,000$

$1 \leq V_i \leq 10^9$

$1 \leq Q \leq 10^{12}$

$1 \leq S_i \leq Q$

Subtask 1 (points: 23)

$N < 10$, $Q < 100$

Subtask 2 (points: 20)

$Q < 10^6$, $V_i < 2000$

Subtask 3 (points: 13)

$V_i = 1$

Subtask 4 (points: 44)

No further restrictions

Examples

standard input	standard output
3 4 5 1 3 -1 0 1 2 2 2	8

Note

In the example, we initially have $5 \times (-1) + 1 \times 0 + 3 \times 1 = -2$ units of power. By switching the thruster with power 3 off and then full forward again, and by switching the thruster of power 5 to full forward, we use all 4 state switches and achieve a forward thrust of $5 \times 1 + 1 \times 0 + 3 \times 1 = 8$. This is maximal.

Problem B. Space Jazz

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Space Team have disguised themselves as the inhabitants of a backwater planet known as "Earth". During their stay they have taken an interest in the music of these so called humans, particularly an obscure genre known as "space jazz". Instead of being played on an ordinary scale, space jazz is played on a scale of 26 notes, labelled 'a' to 'z'. A space jazz composer writes a piece of space jazz in the following way. They start with an empty sheet of paper. They then choose a particular note from 'a' to 'z' and write it down twice. They then repeatedly choose a note (it could be the same or different from the previous) and write it down twice between two adjacent notes. So, for example, a composer might start by writing 'gg', then add 'oo' to make 'goog', then add 'aa' to make 'aagoog', and so on.

The Space Team wants to write down some space jazz so they can try playing it themselves. The problem is that all the performances they listen to leave out notes. Given the notes played in a performance of space jazz, help them figure out the minimum number of notes that were left out, given that the original piece was a valid space jazz composition.

Input

A single line containing a string of lower case English letters, indicating the performance to be analysed.

Output

A single integer, the minimum number of notes the performer must have left out.

Constraints

Let N be the length of the string. Then $1 \leq N \leq 500$.

Subtask 1 (points: 14)

For any note, all occurrences of that note in the performance are adjacent to each other.

Subtask 2 (points: 18)

$N \leq 12$

Subtask 3 (points: 18)

$N \leq 40$

Subtask 4 (points: 20)

$N \leq 100$

Subtask 5 (points: 30)

No additional restrictions.

Examples

standard input	standard output
aagog	1

Problem C. Space Mining

Time limit: 3 seconds
Memory limit: 256 megabytes

The Space Team has gone into rare metal asteroid mining. This is a highly competitive business, and the prospecting is particularly ruthless. So ruthless, in fact, that it is now too unsafe for humans, and robots are used to survey potential asteroids.

The Team has found that whenever they get to an asteroid, a rival company is already there. Regulations for asteroid mining are ad hoc and primitive. Each company places flags on the asteroids they want to mine. After some time, the company with the most flags on the asteroid is awarded the mining rights. As a result, prospecting robots have become rather advanced. To gain an edge, the Team has added weapons to their robots. The rivals, however, responded in kind. Now the Team needs a better strategy for their robots.

Each asteroid is a N by M grid of "blocks" with some blocks accessible, and some inaccessible. The asteroid is symmetric under rotation by 180°. The Team's robot (the robot under your control) is placed at position (S_x, S_y) (that is, in the S_x-th column and the S_y-th row) and your opponent at position $(N - S_x + 1, M - S_y + 1)$.

Every turn, both robots will move exactly one block North, South, East or West. They will uproot whatever flag is in the block into which they move, and plant their own. In addition, they will possibly lay a mine in that block. The mine will detonate whenever a robot next arrives on the block (you detonate your own mine if you run over it!). However, each robot only has Q mines to use on the asteroid.

Despite being low gravity, remote controlled, armored, mine-laying, rare-metal detecting asteroid prospecting machines, the robots are actually pretty delicate, and if they run into each other, they will break down. The robots have collided if either both robots occupy the same block at the same time or the robots exchange positions in one turn (that is if the Team's robot is on block A and the competition is on block B and after a turn the opponents are on A and the Team are on B). Note that both robots move simultaneously.

The Authorities periodically sweep the asteroids, taking bribes and assigning mining rights. After T turns, the Authorities will tally the number of flags and award the contract. If either robot explodes however, the Authorities will detect the shockwave and immediately converge on the asteroid to award the rights.

Make sure this asteroid goes to the Team!

Input

This is an interactive task. You must implement three functions, `init`, `opponentMove` and `move`.

The function `init` is called once at the beginning of the execution. It must have form

- c++: `void init(int N, int M, int Q, int T, std::string[] board, int Sx, int Sy, int subtask);`
- java: `static void init(int N, int M, int Q, int T, String[] board, int Sx, int Sy, int subtask)`
- python: `def init(N, M, Q, T, board, Sx, Sy, subtask)`

This specifies the parameters described above. `board` is an array of strings describing the asteroid, with

corresponding character . if the block is accessible and # otherwise. The subtask specifies which opponent you have (see the subtask sections).

Thereafter, the grader will call `move` and `opponentMove` alternately. The function `move` is defined as

- c++: `std::string move();`
- java: `static String move()`
- python: `def move()`

and must return your action of that turn. Each action you supply must be of the form `(N|S|E|W)B?`, that is, one of `N`, `S`, `E` or `W` indicating which direction you move in, followed optionally by a `B` to indicate that you wish to lay a mine.

If your program makes an illegal action it will score zero. An illegal action is an action that is not of the form described above, a movement off the edge of the asteroid, or a movement into a block that is inaccessible.

The function `opponentMove` is defined as

- c++: `void opponentMove(std::string movement);`
- java: `static void opponentMove(String movement)`
- python: `def opponentMove(movement)`

The string `movement` will be a single letter: the direction in which your opponent moved. You will not be told whether or not they laid a mine in that turn.

Note: If submitting with java, your functions must be wrapped in a class called `mining`.

Constraints

$N, M \leq 50$

$T \leq 4MN$

$Q \leq 10$

Testing

You may test your program using the grader/visualiser provided. To do this, you must first compile your program with the grader:

- c++: `g++ mining.cpp grader.cpp -o mining -std=c++11`
- java: `javac mining.java grader.java`
- python: No action required, but your solution must be in a file named `mining.py`.

You may then run your program by running

`./run.sh <inputfile> <opponentName> <c++|java|py> [sleepTime]`

Here `<inputfile>` is the name of some file describing the asteroid. This is defined below. We have provided you with a sample, `1.in`.

The parameter `opponentName` is the name of one of the provided opponents. The options are `randomPlayer`, `pathPlayer` and `followPlayer`, which are provided for you. Note that you may select your opponent independently of the test case number in the `inputfile`.

The third parameter is whether to run your c++, java or python solution.

The final parameter is optional and specifies the number of milliseconds for the visualiser to sleep between frames. It defaults to 500.

The format of the input file must obey the following:

The first line of the file must contain five integers, N, M, Q, T and the subtask number (in the range 1 to 3 inclusive. This value is ignored.).

The next N lines will describe the asteroid. Each line must contain M characters. This describes a map of the asteroid with a `.` representing an accessible block and a `#` representing an inaccessible block.

The final line of initial must will contain two integers, S_x and S_y.

Example

```
4 4 0 4 1
#...
..#.
.#..
...#
2 2
```

Note

Scoring per game will be as follows. If there are S Space Team flags planted by the end of the game and P opponent flags, and R accessible blocks, the run will score the closest integer within the range $[0, 100]$ to

$$A\left(\left[\frac{S-P}{R}\right]^{0.8}+0.5D\right)$$

where A is 1.2 if the Team's robot is alive (did not collide into the opponent or a mine) at the end and 0 otherwise and D is 0 if the opponent is alive and 1 otherwise.

The score for a subtask will be given by the total number of points available for that subtask, multiplied by the average score per game in that subtask.

Subtask 1 (points: 52)

In this subtask, your opponent will move randomly. To be precise: your opponent will choose a random direction that does not move it off the asteroid, onto an inaccessible block or onto one of its own mines. It tries to lay Q mines randomly during the T turns.

This opponent is implemented in `randomPlayer`.

Subtask 2 (points: 31)

In this subtask, your opponent will move as follows. It will compute the shortest valid path to a block that does not belong to it and move along that path (given the movement will not run it onto one of its own mines). Ties will be broken randomly.

Up to Q mines will be laid on every timestep that is divisible by 5 if the player is within a box of side length 10 from the opponent in that timestep.

This opponent is implemented in `pathPlayer`.

Subtask 3 (points: 17)

In this subtask, your opponent will move so as to follow you. That is, it will find the shortest valid path to the player, and will move along that path. Ties will be broken randomly. It will not lay any mines.

This opponent is implemented in `followPlayer`.