Bloomberg CODECON

Memory Limit: 1350 MB
Time Limit: 5 s

# License to Hack (150 points)

## Introduction

MI6 and CIA have been trying to intercept terrorist communications, but found that all their messages are transmitted in a very cryptic way. 007 was called to the rescue to crack the code. 007 has figured out the algorithm the terrorists have used to encode the message, but needs our help to code in the reverse and find the actual message from the encryption.

The encoding Algorithm figured out by 007 is as follows:

Given a message of any length and a positive integer N, reverse N characters and skip N characters until the end of the string to generate the encrypted message.

## Input Specifications

There are 2 lines of input. The first line is the encrypted message string, which has valid ASCII characters. The second line is a positive integer N.

## Output Specifications

The output is the decrypted message.

## Sample Input/Output

### Input

```
This should be easy!
1
```

### Output

```
This should be easy!
```

### Explanation

If you exchange every letter with itself, the sentence is unaltered.

### Input

```
ihTs suohld  ebeas!y
3
```

### Output

```
This should be easy!
```

### Explanation

"ihT" -> "Thi", "s s" stays the same, "uoh" -> "hou", and so on

## Input

```
!ysae eb dluohs sihT
30
```

## Output

```
This should be easy!
```

## Explanation

30 is larger than the length of the input string, 20, so the output is just the reverse

Memory Limit: 1350 MB
Time Limit: 5 s

# Zipline Hills (250 points)

## Introduction

You operate a ziplining company and are setting up a new route on an ordered series of hills. (A 'zipline route' is a set of ziplines you can ride in order without hiking from hill to hill. For example, taking a zipline from hill A to B, and then taking another zipline from hill B to C, would be a route of length two.) As everyone knows, longer zipline routes are more fun! You want to know the length of the zipline route with the most individual ziplines possible.

You know the height of each hill. Ziplines can only travel from one hill to another hill of lower height. Additionally, your ropes are only long enough to go at most two hills over. For example, from the fourth hill, you can set up a zipline going to the second, third, fifth, or sixth hills.

Given a particular order of hill heights, print out the number of separate zipline connections in the longest route that meets the above constraints.

## Input Specifications

The first line of input will contain a single integer **N**, the number of hills in the test case **(1 ≤ N ≤ 100)**.

That line will be followed by **N** lines, each containing a single integer, representing the sequence of hill heights **(0 ≤ h ≤ 100)** in the test case.

## Output Specifications

Based on the input, print to stdout a single integer indicating the number of hill-to-hill zipline connections in the route with the most possible such connections.

## Sample Input/Output

### Input

```
6
80
24
36
53
91
17
```

### Output

```
3
```

### Explanation

The longest route possible has three zipline connections: 91 → 53 → 36 → 24.

### Input

```
6
67
40
21
9
37
77
```

### Output

```
3
```

### Explanation

Two different routes have length three: 77 → 37 → 21 → 9, and 67 → 40 → 21 → 9.

Note that the longer chain 67 → 40 → 37 → 21 → 9 is **not** a solution, because 40 → 37, three hills apart, is farther than the maximum allowed gap of two hills.

Bloomberg CODECON

Memory Limit: 1350 MB
Time Limit: 5 s

# Anagram Count (250 points)

## Introduction

Tarquin wants to learn a new language but is having trouble deciding on which one. He also likes word games involving anagrams. So he wants to know the count of anagrams for a given language.

## Input Specifications

Your program will take:

- **N**: On the first line by itself, is the first input indicating number of words in the dictionary.
- N lines, each containing one word (it can be assumed there will be no duplicates).

## Output Specifications

On a single line, the number of anagrams in the set.

Notes:

- An anagram is any single word which has the same letters in the same frequencies as another word from the dictionary.

## Sample Input/Output

### Input

```
5
Spare
Spear
Python
Parse
Ear
```

### Output

```
3
```

### Explanation

Spare, Spear and Parse are anagrams of each other

---

### Input

```
4
Agree
Eager
Bin
Nib
```

## Output

```
4
```

## Explanation

Agree and Eager are anagrams of each other. Bin and Nib are anagrams of each other.

---

## Input

```
2
Apple
Plea
```

## Output

```
0
```

## Explanation

Apple and Plea contain the same characters however Apple has an additional P.

Bloomberg
CODECON

# Chemistry 101 (250 points)

## Introduction

The freshman class of DEADBEEF is taking *Introduction to Chemistry*. To prevent the danger of blowing up the class, each page contains the list of chemicals that *must* be in the test tube beforehand to prevent a dangerous chemical reaction. If ANY of the chemicals are added in the **wrong** order, you might blow up the class. As the smartest one in the class, you want to validate the order of chemicals that your classmates are adding to make sure that they will be safe.

You are given the name of a chemical, followed by the number of prerequisites for that chemical, and then the names of those prerequisites.

For example, assume the first page contains `Arraylium 2 Bronyx Chaotium`; the second page contains `Chaotium 2 Bronyx Dopralt`; and the third page contains `Bronyx 1 Dopralt`. If your classmate adds the ingredients in the order of `Dopralt Bronyx Chaotium Arraylium` he is `SAFE!`. If they add the chemicals in any other order, they will blow up the whole class - `BOOM!`.

Please verify your classmates are not going to blow up the class.

## Input Specifications

Your program will take from **STDIN**

- The first line will be **n**, the number of pages in the book.
- The next **n** lines will give a chemical followed by the number of ingredients it requires beforehand, followed by that list of chemical ingredients. Note that chemical names will not contain whitespace, but may contain any other printable ASCII character. For example, `A 3 B C D` indicates that chemicals B, C, and D are prerequisites for chemical A *(that is, all 3 of B, C, & D will need to be added first before A can be added)*
- The last line will indicate the order of ingredients your classmate plans to enter. For example, `X Y Z` means they are adding chemicals in the order of X followed by Y followed by Z.

## Output Specifications

The output should be `SAFE!` or `BOOM!`. `SAFE!` indicates that the order is valid. `BOOM!` indicates the the order caused a chemical reaction dangerous for the students!

## Sample Input/Output

### Input

```
2
X 2 Y Z
Y 1 Z
Z Y X
```

### Output

```
SAFE!
```

## Explanation

Since you need to add Y & Z before X and Z before Y, Z->Y->X is a safe chemical order

---

## Input

```
3
Arraylium 2 Bronyx Chaotium
Chaotium 2 Bronyx Dopralt
Bronyx 1 Dopralt
Dopralt Bronyx Arraylium
```

## Output

```
BOOM!
```

## Explanation

You blew up the class because Choatium needs to be added before Arraylium!

Bloomberg CODECON

Memory Limit: 1350 MB
Time Limit: 5 s

# Robo-Golfer (250 points)

## Introduction

After watching a golf game on TV, you drive to a golf course with some friends to try your luck at golf. Unfortunately it doesn't go so well. Fortunately, you are an engineer, and you decide to create a robot that plays golf for you.

This robot will hold a set of golf clubs, each being capable of hitting the ball an exact distance in yards. The robot will select a club and hit the ball towards the hole. If the hole is overshot, the robot will turn around and keep shooting towards the hole.

You decide to test if the set of clubs you select will be able to drive the ball into the holes on your favorite golf course.

## Input Specifications

The first line will be an integer 1 <= N <= 10, representing the number of golf clubs. The second line will list the distances (1 <= X <= 300) that each club will hit the ball, separated by a space.

The third line will be an integer 1 <= M <= 50, representing the number of holes your course has. The fourth line will list the distances (1 <= Y <= 600) from each tee (starting shot) to hole (goal), separated by a space.

## Output Specifications

For each integer M, a line containing yes or no will be listed, representing if it is possible to sink the ball into each hole using the given set of golf clubs.

## Sample Input/Output

### Input

```
2
10 30
3
40 20 15
```

### Output

```
yes
yes
no
```

### Explanation

1. 40 = 10 + 30

2. 20 = 30 - 10

15 cannot be reached with 10 and 30

### Input

```
2
8 14
3
30 6 51
```

### Output

```
yes
yes
no
```

### Explanation

1. 30 = 14 + 8 + 8

2. 6 = 14 - 8

51 cannot be reached with 8 and 14

Bloomberg CODECON

Memory Limit: 1350 MB
Time Limit: 5 s

# The Bedlam in the Pantry (400 points)

## Introduction

Fenwick works at Bloomberg. At lunchtime, Bloomberg offices provide soups & salads. There are usually several spots where the food is distributed, but the pantry is crowded by other people also trying to get free lunch.

Fenwick is not the type of person that waits in line. He wants to know the shortest distance he'd have to cross to sneak up to where the food is distributed.

## Input Specifications

First line: N and M, the dimensions of the pantry (4 <= N <= 10000, 4 <= M <= 10000).

Other lines: N lines of M characters, representing the top-down map of Bloomberg's pantry. 'X' represents an obstacle, '.' represents free space, '*' represents Fenwick, 'S' represents lunch foods.

The outer perimeter of the pantry will always consist of obstacles, representing people closing-in onto the food.

Fenwick can only move up, down, left or right, because he bought a faulty hoverboard.

## Output Specifications

A single integer, telling the length of the shortest path to any food. Output -1 if no foods are reacheable.

## Sample Input/Output

### Input

```
6 11
XXXXXXXXXXX
X.S.X.....X
X.XX...*..X
X....XXXX.X
XS....S...X
XXXXXXXXXXX
```

### Output

```
7
```

### Explanation

There are two paths of length 7 that lead to bottom-right food. The other foods are further away.

---

### Input

```
6 11
```

```
XXXXXXXXXXX
X.S.X.....X
X.XX...*..X
X...XXXXXXX
XS....S...X
XXXXXXXXXXX
```

### Output

```
-1
```

### Explanation

No foods are reacheable without queueing.

Bloomberg CODECON

Memory Limit: 1350 MB
Time Limit: 5 s

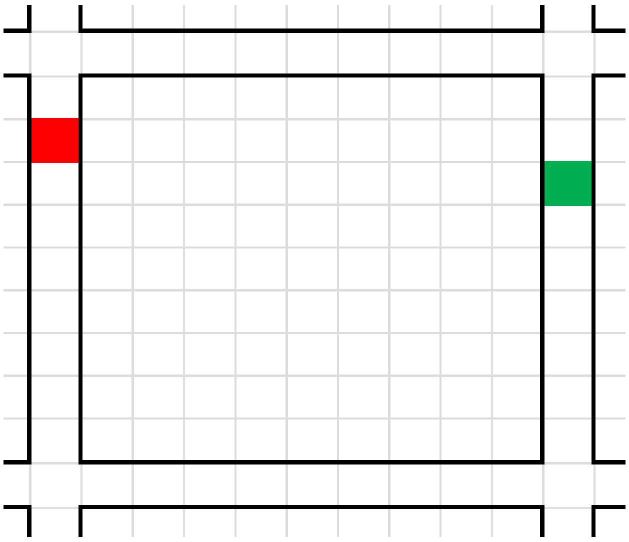
# Zombie Ambulance (600 points)

## Introduction

You're an ambulance driver in a city hit with a zombie plague. Patients are scattered across the city, and you're tasked with bringing back as many as you can before time runs out and they join the living dead!

Your ambulance starts at a specified **hospital location** on a coordinate grid. To save a particular patient, you have to drive to them and then return to the hospital. The ambulance travels at a constant 1 unit/sec, so **the time to rescue a particular patient is twice the distance of the optimal path from the hospital to that patient.**

However, there's a constraint: **your ambulance must drive around city blocks, not through them.** The city is made up of square 10x10 blocks on a coordinate grid, with streets along `x = 0, y = 0, x = 10, y = 10,` etc. This means the travel distance between two points is sometimes not just the Manhattan distance. In this example, the distance between the red and green tiles isn't the Manhattan distance of 11, but instead 15, to drive around the block:



Your task is to determine the maximum number of patients it's possible to save in the allotted time, and identify which patients are saved.

- Both the hospital and all the patients are guaranteed to be on a street (at least one of x and y is 0 mod 10).
- Loading, unloading, and curing patients all happen instantly.
- Driving through the hospital is fine.
- You can only carry one patient at a time - no carpools!
- For certain inputs, multiple solutions are possible, but **you will only be given test cases with a unique solution**.

## Input Specifications

Your program will take one space-separated line with problem parameters, followed by one line for each patient.

The first line will contain, separated by spaces:

- **X**, the integer x-coordinate of the hospital **(0 ≤ X ≤ 100)**

- **Y**, the integer y-coordinate of the hospital **(0 ≤ Y ≤ 100)**
- **T**, the amount of time you have to rescue patients, in seconds **(0 ≤ T ≤ 2000)**
- **N**, the number of patients present **(0 ≤ N ≤ 20)**

The first line will be followed by **N** additional space-separated lines, containing:

- a single unique capital letter labeling the patient
- **X**, the integer x-coordinate of the patient **(0 ≤ X ≤ 100)**
- **Y**, the integer y-coordinate of the patient **(0 ≤ Y ≤ 100)**

## Output Specifications

Based on the input, print to stdout a single line containing:

- the maximum number of patients it's possible to rescue in the **T** seconds provided
- if at least one patient can be saved, the labels of the patients that will be saved, sorted in ascending alphabetical order, separated by spaces

## Sample Input/Output

### Input

```
10 10 40 1
A 20 20
```

### Output

```
1 A
```

### Explanation

Patient A, the only patient, can be rescued in exactly 40 seconds:

1. (10,10) to (20,10) (ten seconds)
2. (20,10) to (20,20), pick up patient (ten seconds)
3. (20,20) to (20,10) (ten seconds)
4. (20,10) to (10,10), drop off patient (ten seconds)

---

### Input

```
17 10 60 4
A 0 1
B 12 10
C 30 6
D 38 20
```

### Output

```
2 B C
```

### Explanation

Patient B can be rescued in 10 seconds, and Patient C in 34 seconds. Patient A could be rescued in 54 seconds, but it would then be impossible to rescue a second patient. There is no other pair of patients that can be rescued with the 60 seconds provided.

Memory Limit: 1350 MB
Time Limit: 5 s

# Ore Mining (800 points)

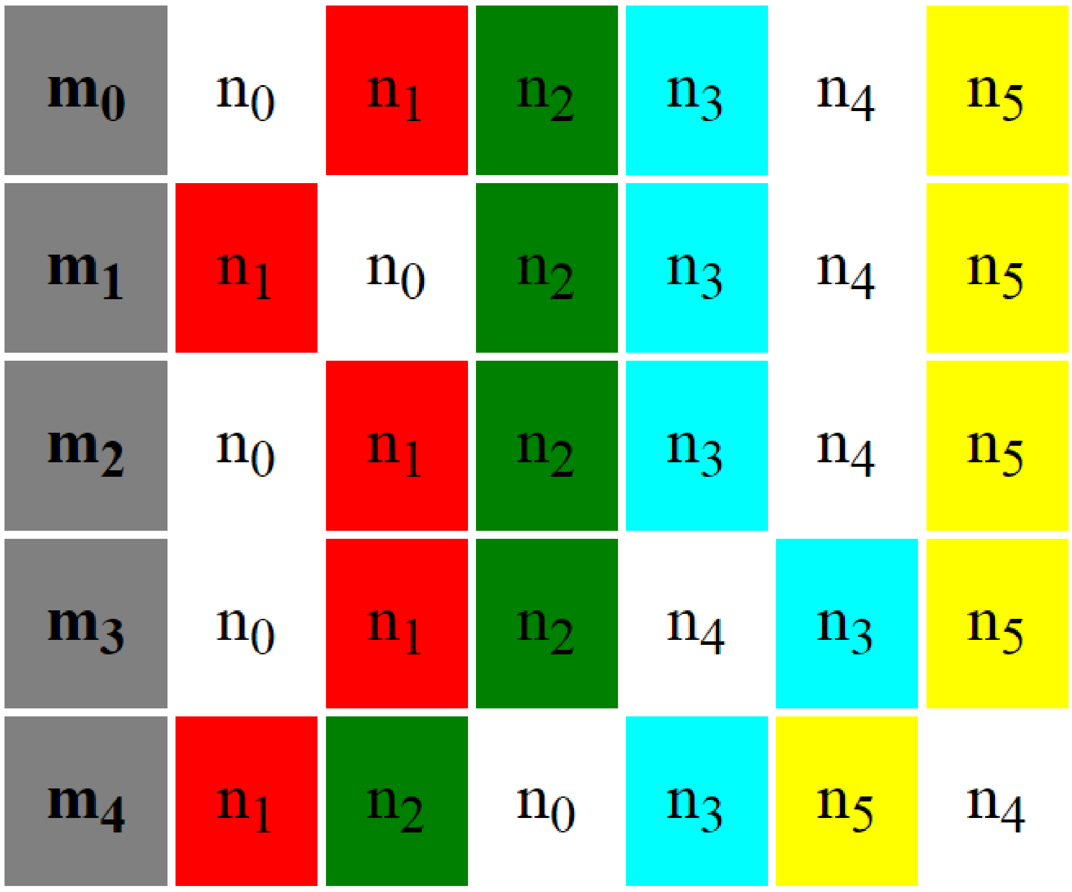
## Introduction

A mining operation has discovered deposits of N different kinds of liquid ore, which they have labeled $n_i$ for $0 \le i < N$. The deposits are spread throughout M sheets of rock, which have been labeled $m_j$ for $0 \le j < M$.

To simplify the problem, let us imagine we are looking at a cross section of the rock which has been divided into N columns and M rows, for a total of N*M cells. Each type of ore $n_i$ appears exactly once in each sheet of rock $m_j$ (i.e. once per row).

The crew needs to drill paths from the top layer $m_0$ to the bottom layer $m_{M-1}$ such that each path connects deposits of one type of ore $n_i$ across all M layers.

In the example below with N=6 and M=5, you can see the best option is to drill 4 paths, for ores $n_1$, $n_2$, $n_3$, and $n_5$.



$m_0$ $n_0$ $n_1$ $n_2$ $n_3$ $n_4$ $n_5$
$m_1$ $n_1$ $n_0$ $n_2$ $n_3$ $n_4$ $n_5$
$m_2$ $n_0$ $n_1$ $n_2$ $n_3$ $n_4$ $n_5$
$m_3$ $n_0$ $n_1$ $n_2$ $n_4$ $n_3$ $n_5$
$m_4$ $n_1$ $n_2$ $n_0$ $n_3$ $n_5$ $n_4$

Let us define a valid path for ore $n_i$ as M-1 line segments, each of which connects a deposit of ore $n_i$ in layer $m_j$ to the deposit of ore $n_i$ in layer $m_{j+1}$, such that no line segment intersects the line segment of another path. There should be no more than 1 path per type of ore.

What is the **maximum number of valid paths** that can be drilled by the above definition?

## Input Specifications

The first line will contain two space-seperated integers, N and M, where $1 \le N < 1000$ and $2 \le M < 1000$.

The next M lines will contain N space-seperated, case-sensitive names of ore types (all names are 20 alphabetic characters or fewer). The same N names will appear on each line, although possibly not necessarily in the same order.

## Output Specifications

The maximum number of paths which can be drilled and considered valid by the above criteria.

## Sample Input/Output

### Input

```
3 2
A B C
A C B
```

### Output

```
2
```

### Explanation

Here we have two ways to drill 2 paths: {A, B} or {A, C}.

---

### Input

```
3 2
Alpha Beta Gamma
Alpha Beta Gamma
```

### Output

```
3
```

### Explanation

Here we have one way to drill 3 paths: {Alpha, Beta, Gamma}.

---

### Input

```
3 3
A B C
B C A
C A B
```

### Output

```
1
```

### Explanation

Here we have three ways to drill 1 path: {A} or {B} or {C}.

---

### Input

```
5 3
A B C D E
A C E B D
A B D C E
```

### Output

```
3
```

### Explanation

Here we have two ways to drill 3 paths: {A, B, D} or {A, C, E}.