Spring 2026 UMW Programming Contest

Overview

Welcome to the Spring 2026 UMW Programming Competition! This is a contest wherein you compete amongst your fellow students, alumni, and professors in an effort to solve a set of programming problems as fast as possible.

Problems

This packet contains a set of problems in (more or less) increasing difficulty. The team who solves the most problems is declared the winner. In the event of a tie, the team with the best time wins. The best time also includes penalties for wrong submissions. Each problem has a general description, details on the formats for input and output, and example input and output.

Input is done from standard input and output is done to standard output. *Do not open any files inside of your program.* Output of your program must match that of the correct output *exactly*. You can solve the problems in C, C++, Java, Python, or Rust. Up to date versions of these languages will be used to judge your entries. You should save your code using the standard extension for your language of choice (.c .cpp .java .py and .rs respectively).

Rules

You are allowed to have any printed material such as books or printouts. You are also allowed to access standard library documentation for your language of choice. *No other use of the internet is permitted.* You are also not allowed to copy and paste code from any source be it the web, a thumb drive, or another file that you have. AI coding assistants are not allowed. Only one computer is allowed to a team.

Logging In

To log in to the system, direct your browser to `http://34.59.95.151/`. You will then need to enter your team name and password which are provided for you. This will bring up the interface for you to submit solutions to the problems, ask for clarifications, and see how your team is doing.

Submitting

To submit a solution, click the green "Submit" button in the upper right. Browse for the source file you want to submit, and select the problem you are solving and the language. Click Submit again, and the program will be submitted for judging. The result will show as pending while the program is being judged, but after a few moments will be marked as correct or incorrect.

DO NOT TURN OVER THIS SHEET UNTIL THE CONTEST STARTS

Problem A: Penguin Sledding

Aang has just arrived at the Southern Water Tribe and is desperate to go penguin sledding! Katara explains that the otter-penguins are very skittish. They will only let you ride them if you offer them a tasty fish first.

Aang has been searching the ice chunks all morning. Write a program to determine if Aang gets to go sledding based on how many fish he found.

Input

The input consists of a single line containing one integer, F, representing the number of fish Aang has found.

Output

If Aang has at least 1 fish ($F > 0$), output the string `SLEDDING!`. If Aang did not find any fish ($F = 0$), output the string `NOT TODAY`.

Sample Input 1

```
3
```

Sample Output 1

```
SLEDDING!
```

Sample Input 2

```
0
```

Sample Output 2

```
NOT TODAY
```

Problem B: Momo's Moon Peaches

Aang just found a bush full of delicious moon peaches and gave a whole pile of them to Momo. However, Sokka is also incredibly hungry. Every time Momo turns his head to look at a butterfly, Sokka sneaks a few peaches from the pile!

Given the number of moon peaches Aang originally gave Momo, and the number of peaches Sokka managed to sneak away, calculate how many moon peaches Momo actually gets to eat.

Input

The input consists of a single line containing two space-separated integers: M, the number of moon peaches Aang gave Momo, and S, the number of peaches Sokka snuck away. You can assume M will always be greater than or equal to S.

Output

Output a single integer representing the number of moon peaches Momo has left.

Sample Input 1

```
10 3
```

Sample Output 1

```
7
```

Sample Input 2

```
5 5
```

Sample Output 2

```
0
```

Problem C: My Cabbages!

Cai, a cabbage salesman in the Earth Kingdom, has terrible luck with his cabbage cart being routinely destroyed. Each time that it's destroyed, he loses half of his cabbages, rounded down. A lost cabbage is one that is smashed and no longer able to be sold. When he purchases cabbages from his supplier, he always buys 15 at a time.

Given a list of cabbage events, either a purchase of 15 cabbages, or half of his stock being destroyed, you should compute how many cabbages he has left. Cai always starts with zero cabbages and never actually sells any.

Input

The first line of input is an integer, N, giving the number of cabbage events in the input. Following that are N lines, each giving one cabbage event which is either the word "purchase" or the word "smash!".

Output

You should output a single integer, giving the number of cabbages that are left at the end.

Sample Input

```
6
purchase
purchase
smash!
purchase
smash!
smash!
```

Sample Output

```
8
```

Problem D: Packing Appa

Team Avatar is preparing to leave the Northern Water Tribe. Sokka is in charge of packing Appa, the flying bison, with essential supplies. Appa is incredibly strong, but he has a strict maximum weight limit for taking off safely.

Sokka has laid out a variety of items (tents, water skins, sleeping bags, boomerang polish, etc.) and wants to pack as many *individual items* as possible so the team feels well-prepared, regardless of what the items actually are.

Write a program to determine the maximum number of items Sokka can load onto Appa without exceeding the bison's weight limit.

Input

The first line of input contains two space-separated integers: W, Appa's maximum weight limit, and I, the total number of items Sokka has laid out. The second line contains I space-separated integers, representing the weight of each individual item.

Output

You should output a single integer giving the maximum number of items that can be packed onto Appa.

Sample Input 1

```
50 7
15 5 25 10 12 8 20
```

Sample Output 1

```
5
```

Sample Input 2

```
10 3
15 20 12
```

Sample Output 2

```
0
```

Problem E: Iroh's Tea Shop

Iroh is running a tea shop in Ba Sing Se with the help of his nephew Zuko. He keeps an inventory of all the tea ingredients that he has in stock. For example, he might have 7 units of ginger root and 10 units of white jasmine. He also has recipes of the tea blends that he sells, many of which are of his own invention. For each of these recipes, he has listed out how many units of which ingredients go into the recipe.

He would like a way of knowing how many servings of a recipe he can sell, based on the available stock of ingredients. He can only prepare a serving based off a recipe if he has enough of all the ingredients that recipe contains. You should write a program to help him in this task.

Input

The first line of input contains an integer I, which gives the number of ingredients in stock in the shop. Following that are I lines, each containing an integer, giving the quantity, and the name of the ingredient.

Following these lines is a blank line, and then an integer, R, which is the number of ingredients in the recipe. Following that are R lines, each containing the number of units of that ingredient needed, and the name of the ingredient.

Output

You should output a single integer giving the number of servings of tea that can be made from the recipe given.

Sample Input 1

```
6
7 ginger root
10 white jasmine
21 ginseng
18 peppermint
4 fire lily root
11 dragon blossom

3
1 ginger root
3 white jasmine
2 dragon blossom
```

Sample Output 1

```
3
```

Sample Input 2

```
6
7 ginger root
10 white jasmine
21 ginseng
18 peppermint
4 fire lily root
1 dragon blossom

3
1 ginger root
3 white jasmine
2 dragon blossom
```

Sample Output 2

```
0
```

Problem F: Avatar Lifetimes

The Avatar is the human embodiment of the Avatar spirit, and the only person who can bend all four elements. When the Avatar dies, this spirit is reborn in a new Avatar, continuing the cycle. There is always exactly one Avatar at a time. There have been many Avatars over the years, with the most recent ones (such as Korra, Aang, Roku, and Kyoshi) having widely known dates of birth and death. The farther back in time we go, however, the more uncertain the identity of the Avatar becomes.

There are some mythical figures who are believed to have been powerful benders and *may* have been incarnations of the Avatar. Professor Zei is doing historical research into the Avatar cycle and would like help writing a program to determine if certain mythical figures may or may not have been an Avatar. If a figure's lifetime overlaps with that of a known Avatar, it's impossible for that figure themselves to be an Avatar (because only one exists at any time). If the figure's lifetime does not overlap any known Avatar's, it is possible they are an Avatar.

Years in the Four Nations are divided into years before the Air Nomad Genocide (BG) and years after this event (AG). 0 AG is the year this event occurred and the Hundred Year War began. Years ending in BG count backwards from this event and those ending in AG count forwards from it. So for instance Avatar Aang was born in 12 BG, the war began when he was 12 in 0 AG, and then he died in 153 AG when he was 165.

There is no conflict with one Avatar being born in the same year that the previous Avatar died, as the re-incarnation happens immediately.

Input

The first line of input contains an integer K, giving the number of known Avatars. Following that are K lines, each containing two years and the name of the Avatar, separated with commas. Each year contains an integer and either "BG" or "AG", separated with a space. The first year is the Avatar's birth year and the second is their death year. No input will have a death year which is before a birth year. No known Avatar will have a lifespan overlapping that of another known Avatar.

Following these lines is a blank line, followed by a line of information for the mythical figure in question. That line will contain the birth and death years and name, in the same format as described above.

Output

If the mythical figure's lifetime does not overlap that of any known avatar, you should output "No overlap". Otherwise you should print a line saying "Overlap: NAME" (replacing NAME with the name of the Avatar the figure's lifetime overlaps with). If the lifetime overlaps that of multiple Avatars, you should have such a line for each one, with them appearing in the same order as given to you in the input.

Sample Input 1

```
5
12 BG,153 AG,Aang
82 BG,12 BG,Roku
312 BG,82 BG,Kyoshi
415 BG,345 BG,Yangchen
643 BG,560 BG,Szeto

560 BG,517 BG,Salai
```

Sample Output 1

```
No overlap
```

Sample Input 2

```
5
12 BG,153 AG,Aang
82 BG,12 BG,Roku
312 BG,82 BG,Kyoshi
415 BG,345 BG,Yangchen
643 BG,560 BG,Szeto

362 BG,311 BG,Zalir
```

Sample Output 2

```
Overlap: Kyoshi
Overlap: Yangchen
```

Problem G: Combustion Man

Prince Zuko hires a fire-bending assassin to track down and eliminate Avatar Aang. This assassin is known to the protagonists only as "Combustion Man" as they never learn his name. Combustion Man has the unusual ability to fire a laser of fire out of a tattoo of a third eye on his forehead.

Aang is currently attempting to evade Combustion Man in a small town, which provides some number of obstacles he can use to avoid entering the path of Combustion Man's deadly weapon. These can be houses, walls, rock formations or anything else that can block the laser attack.

You should write a program to read in the current positions of Aang and Combustion Man, and the locations of all obstacles in the area. These will be given as rectangles. You should then compute whether Aang can be hit by a laser fired directly from Combustion Man to Aang, or whether the laser would hit one of the obstacles instead.

Input

The first line of input contains 2 integers, giving the X and Y coordinates of Aang. The second line contains 2 integers, giving the X and Y coordinates of Combustion Man. The third line contains a single integer O, giving the number of obstacles in the space.

Following that will be O lines, each of which represent one rectangular obstacle, with the first being obstacle 1, the second being obstacle 2, etc. Each obstacle line contains 4 integer values. The first two are the X and Y coordinates of the *upper-left hand* corner of the obstacle, and the second two give the X and Y coordinates of the *bottom-right hand* corner.

All of the coordinate values will be in the range of 0 to 100. These values are all given as input to your program as integers, but the laser of Combustion Man travels directly towards Aang, and its path travels across any real number value.

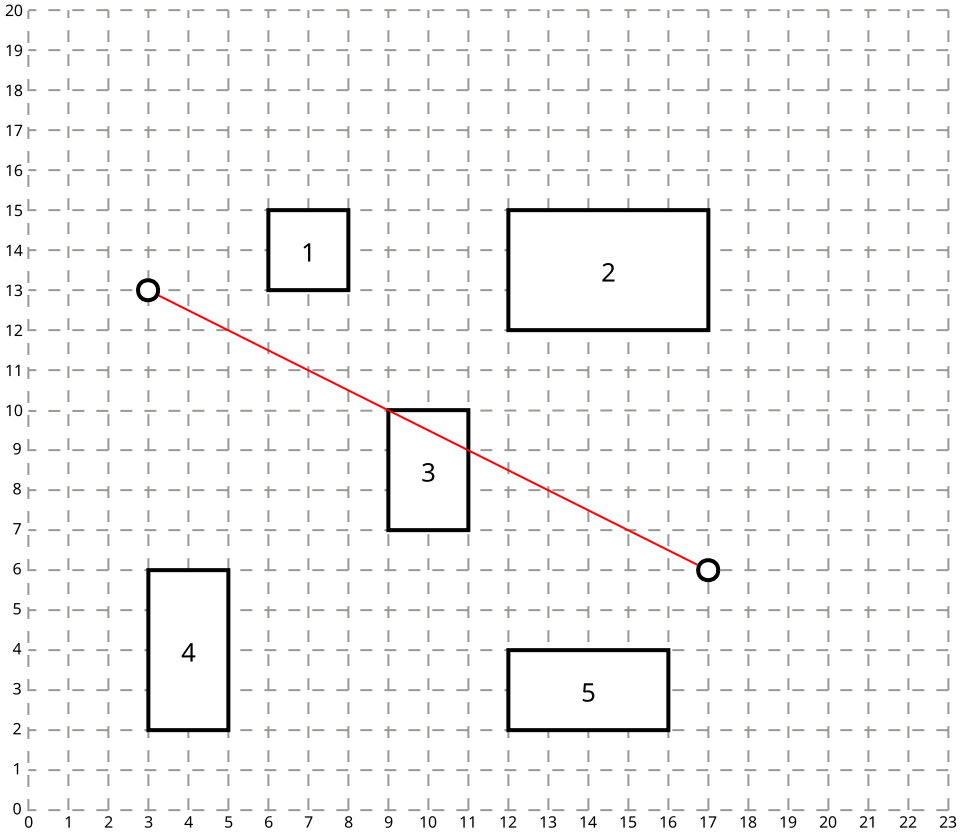
Output

Output should consist of either the line "Aang hit", if the combustion man can hit Aang with his laser, or "Obstacle X hit", replacing 'X' with the number of the obstacle which was hit by the laser instead.

Sample Input

```
17 6
3 13
5
6 15 8 13
12 15 17 12
9 10 11 7
3 6 5 2
12 4 16 2
```

Visualization of Sample Input



Sample Output

```
Obstacle 3 hit
```

Problem H: The Secret Tunnel

Aang and Katara are trapped in the Cave of Two Lovers. The cave is a dark, sprawling labyrinth, and they must navigate from their starting position to the exit.

However, the cave is heavily patrolled by badgermoles. Badgermoles are blind but highly sensitive to vibrations in the earth. If Aang and Katara step into a tunnel directly adjacent (up, down, left, or right) to a badgermole, they will be caught!

Find the minimum number of steps required to reach the exit without stepping on a badgermole or any cell adjacent to one. If there is no safe path, output `SECRET TUNNEL`.

Input

The first line contains two integers, R and C, representing the number of rows and columns in the cave grid. The next R lines contain C characters each, representing the map: `A` represents the starting location, `E` represents the Exit, `.` represents an empty tunnel, `#` represents a solid rock wall, and `B` represents a Badgermole. The characters on a line are separated by space characters.

Output

Output a single integer representing the minimum number of steps to go from `A` to `E`. If the exit cannot be reached safely, output the string `SECRET TUNNEL`.

Sample Input 1

```
5 6
A . . # . E
. # . . . .
. # B # . .
. . . . . .
. . . . . .
```

Sample Output 1

```
13
```

Sample Input 2

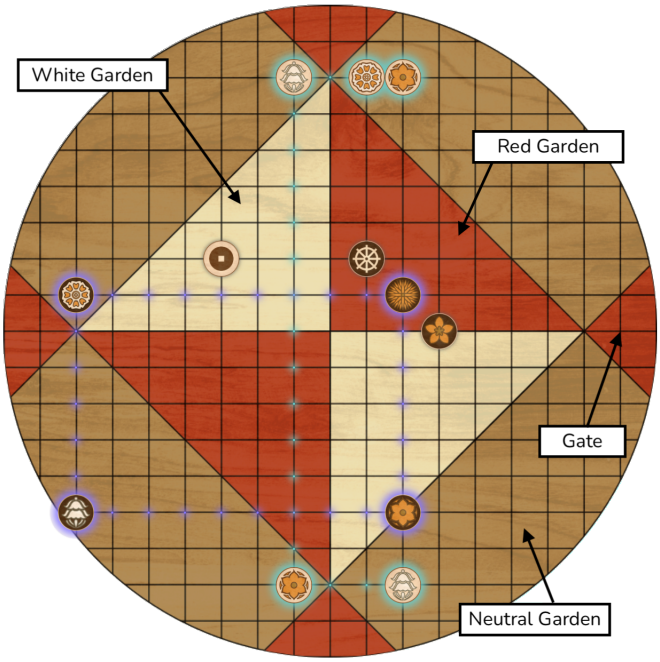
```
4 4
A . . .
. # B #
. # . .
# . E .
```

Sample Output 2

```
SECRET TUNNEL
```

Problem I: Pai Sho

The two-player board game Pai Sho is played throughout the four kingdoms. It uses a circular board on which players place tiles at the intersections of lines. The board is divided into four "gates" (the triangular areas at the edges of the board), two red gardens, two white gardens, and four neutral gardens, as depicted in the following diagram:

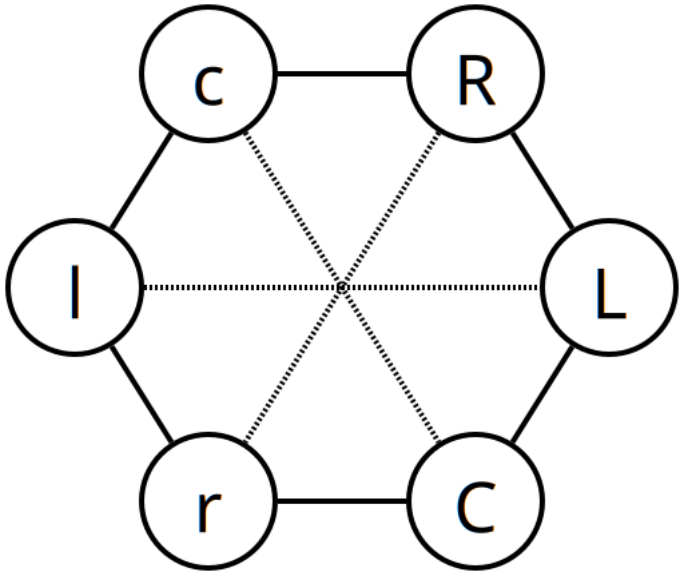


The board consists of 19 rows and 19 columns. However, since it is circular, not every row/column intersection can have a tile placed on it. See the example input to see which points can support tiles being placed on them.

Pai sho is played with one player being white and the other being red. During play, players can choose to either place new tiles on the board, which must begin in one of the gate areas, or move existing tiles. There are different types of tiles, including basic flower tiles. Each player has three types of basic flower tiles: the rose, lily, and chrysanthemum. The white players tiles are represented in the input with lower-case letters and the red player with upper-case ones, as shown in the following table:

	White	Red
Rose	r	R
Lily	l	L
Chrysanthemum	c	C

These basic flower tiles must be moved out of the gates, where they are said to be “growing”, and into the gardens, where they are “blooming”. The six basic flower tiles have relationships among them. There are two types of relationships: harmonies and clashes. The following diagram depicts the harmony and clash relationships between the basic flower tiles:



The solid line connections along the outside represent harmonies. For example, the white rose tile (r) is harmonious with the white lily (l) and the red chrysanthemum (C). The clashes are the dashed lines across the center. Each flower clashes with its counterpart in the opposite color. So the white rose (r) clashes with the red rose (R).

To create a harmony or a clash, the two flower tiles need to share either a row or a column, i.e. they need to be lined up horizontally or vertically, and have no tile in between them. For instance, if a red lily tile is placed in the same row as a red rose tile, with no tiles separating them, they create a harmony.

The goal of the game is to create a **harmony ring** which consists of a chain of harmonies that connect together. For example, the following tiles create a harmony ring:

```
. . . . . . . . . . .
. l . . . . . . . c .
. . . . . . r . . l .
. . . . . . . . . . .
. r . . . . C . . . .
```

This harmony ring loops through six harmonies, returning to where it began. Starting at the upper-left, they are: l-r, r-C, C-r, r-l, l-c, c-l. Note that not all of these tiles are harmonious with each other, but the

ones lined up vertically or horizontally are. The two chrysanthemum tiles do not create a clash because they share neither a row nor a column.

Also note that this harmony ring contains both red and white tiles. This is common and the rule is that the harmony ring belongs to the player who owns the majority of the tiles in the ring, which would be white in this instance. If a harmony ring contains an equal number of red and white tiles, then it belongs to neither player and the game continues.

Furthermore to win, the player's harmony ring must encircle the center point of the Pai Sho board without touching it. There also can be no clashes anywhere on the board. That is, no tiles which clash can share a row or column without tiles separating them.

Pai Sho uses other tiles besides the six basic flower tiles, such as Iroh's favorite white lotus tile. However they have no direct effect on the winner of a game, (except that they can block harmonies/clashes by separating basic flower tiles).

The Ba Sing Se Pai Sho League has asked for your help in more quickly determining winners in Pai Sho games. They want you to write a program to read in the state of a Pai Sho board and determine which player, if any, has won. You can assume the input will contain at most one winning harmony ring.

Input

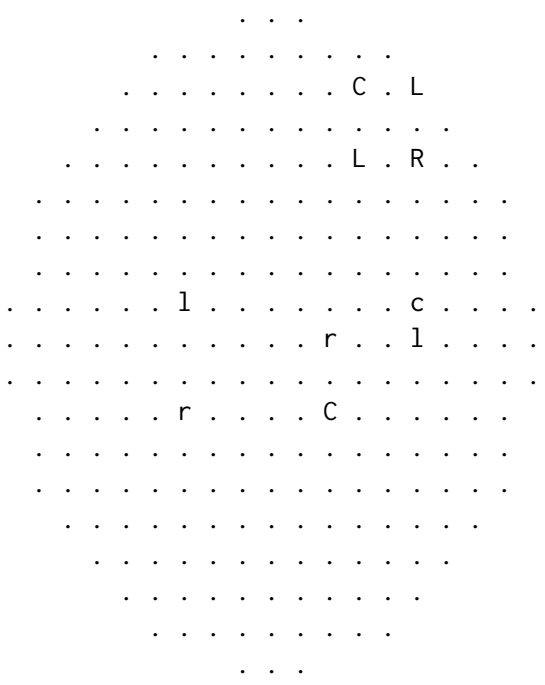
Input consists of 19 lines of text, each corresponding to a row of the Pai Sho board. Each row contains a number of intersection points to form the circular playing area. The input contains leading spaces for the points which are off the left end of the board. For example, the first line contains 8 columns of leading spaces. There are no trailing spaces after each line. There are also spaces between each point, to make the input appear more proportional.

Each intersection point either contains a . character, which indicates that the point is empty, or a lower-case or upper-case letter. These letters refer to the tiles placed by the two players, which include the six characters for basic flower tiles that are detailed above. Other tiles besides those six can block harmonies and clashes, but do not create them.

Output

If one player has created a harmony ring – a chain of harmonies which encircle the center point of the board and which consists of mostly their tiles – and also there are no clashes on the board, then you should output the winner with a line saying either "Red wins!" or "White wins!". If no player has such a harmony ring, or if there is a clash on the board, you should output "No winner.".

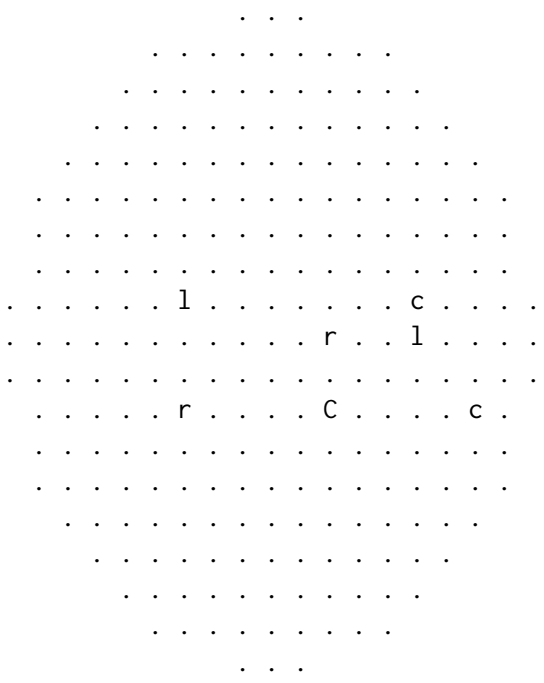
Sample Input 1



Sample Output 1

```
White wins!
```

Sample Input 2



Sample Output 2

```
No winner.
```