SPRING 2025 UMW PROGRAMMING CONTEST

Overview

Welcome to the Spring 2025 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, or Python. 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 and .py 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://34.69.244.12/`. 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: Battery Check

CORA (Computerized Organizational Robotic Assistant) is a robot who works for BotWorks Inc. After a busy day, when the other robots return to their charging stations, she normally waters the plants in the factory. After making her rounds tonight, however, she notices something different. The lights flicker and then go out. The eyes of the other robots in their charging stations turn off for a few moments and then light back up in a sinister red.

Something is definitely not right here. CORA needs to investigate, but first she needs to decide if her battery has enough charge to continue. If her battery is critically low, at less than 10% charge, she will need to return to her charging station. If it's low, which is 10% or more, but less than 30%, she can continue in low-power mode. If it's 30% or more, she can continue as normal.

You need to write a program which will check CORA's current battery level and indicate the battery status as either critically low, low, or normal.

Input

Input will consist of a single line containing a floating-point number, giving the current charge of CORA's battery.

Output

Output should consist of a single line giving the state of the battery as either "Critically low", "Low", or "Normal" corresponding to the three levels in the description.

Sample Input 1

```
6.31
```

Sample Output 1

```
Critically low
```

Sample Input 2

```
54.96
```

Sample Output 2

```
Normal
```

Problem B: Console Distance

Something is definitely not right. The other robots in the factory have begun to run amok, upsetting tables and smashing computers. Now that CORA is sure her battery is in good shape, she plans to investigate. She needs to access the computer system in the factory, and that means navigating to the console on the other side of this room.

Help CORA by writing a program that tells how far away she is from the access console. Her position, and that of the console, will be given as (x, y) coordinates. The distance between two points in space is given by the formula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

You should output the distance between CORA and the console as a number rounded to 2 decimal places.

Input

Input consists of 4 lines of input, each containing a positive integer. The first two lines give the X and Y coordinate of CORA (with X being given first), and the second two lines give the X and Y coordinates of the console (again with X being first).

Output

Output will consist of a single line giving the distance between CORA and the console, rounded to two places to the right of the decimal.

Sample Input 1

```
5
10
24
50
```

Sample Output 1

```
44.28
```

Sample Input 2

```
31
4
6
22
```

Sample Output 2

```
30.81
```

Problem C: Power Surge Detection

CORA navigates the distance to the computer console on the other side of this room. She needs to login to the console to access the factory systems, but the other robots are causing havoc in the electrical grid in the facility.

CORA needs to check if and when a power surge will occur before she can safely access the factory terminal. A power surge occurs when a power level is significantly higher than the previous power level. You need to write a program which will read in a list of power levels and find the first power surge where the power level jumps by 10 or more units from the previous reading, or report that no power surge occurs.

Input

The first line of input will contain an integer N giving the number of power readings in the input. Following that will be N lines, each containing one positive integer representing a power level reading.

Output

Output should either be "Surge at P" where 'P' is the first power level that is at least 10 units higher than the previous one, or "No surge detected" if there is no surge in the input.

Sample Input 1

```
5
78
83
95
92
109
```

Sample Output 1

```
Surge at 95
```

Sample Input 2

```
6
55
58
42
51
60
51
```

Sample Output 2

```
No surge detected
```

Problem D: Log Scanning

Now that CORA has sorted out the power surge issue, she can safely login to the nearest terminal and look for anything suspicious.

She will look in the user authentication logs for the most recent other user to login to the system. This might indicate the party responsible for the strange behavior of the other robots in the facility. She will need to scan the log file of login attempts for the last *successful* login of a username other than `cora`, (because she herself has just logged in successfully).

Input

The first line of input will contain an integer, N, giving the number of lines in the log file. Following that will be N lines. Each of these contains the word "user", a space, the username of the user attempting to login, an ellipsis, and whether the attempt was successful or not. User names will consist of only lower-case letters and digits. There is a space on either side of the ellipsis.

Output

Output should be a single line containing the username of the last successful login attempt, except for CORA herself. If there was no such successful login attempt, simply print "none".

Sample Input 1

```
6
user zlee2 ... success
user asmith ... failure
user asmith ... success
user umbr4 ... success
user eroberts ... failure
user cora ... success
```

Sample Output

```
umbr4
```

Sample Input 2

```
4
user mgarcia ... failure
user fsmith6 ... failure
user pthomas ... failure
user cora ... success
```

Sample Output 2

```
none
```

Problem E: Pin Problems

CORA finds that the user umbr4 was the most recent user to login to the system. This username corresponds to Umbr4, a robot that BotWorks rented from AetherTech, a competing firm. It looks like it was sent in to perform some corporate sabotage! CORA will need to get to the security room on the top floor and set things right.

She heads to the elevator and finds it locked. There is a keypad to type in the four-digit access code. In a curious perversion of security, BotWorks changes the elevator code every week, but leaves a post-it note on the wall with the current code. Unfortunately the ink on the post-it has been smeared and one digit is illegible.

CORA will need to try each of the possible values for the smeared-out digit so she can get the elevator doors open. You need to write a program which will read in the code with the unreadable digit and print all the possible codes it could be. There will always be exactly one digit that can't be read.

Input

Input consists of a single line of input containing four characters. Three of these characters will be digits 0-9 and the other will be the underscore character which indicates the illegible digit. The underscore can appear in any of the four character positions.

Output

Output should consist of each possible access code. They should be printed on individual lines, in ascending order.

Sample Input

```
6_73
```

Sample Output

```
6073
6173
6273
6373
6473
6573
6673
6773
6873
6973
```

Problem F: Weight Limits

After trying a few of the possible pins, CORA hits on the right one and the elevator doors slide open. She steps into the elevator but finds that it has already got a number of boxes loaded in it. The elevator has a strict weight limit and CORA will likely need to move some of these boxes out before it can take her up.

CORA takes the same amount of time to move a box out of the elevator regardless of how much the box weighs. She also wants to move as few boxes out as possible, as time is of the essence. You need to write a program to determine the minimum number of boxes that need to be moved out so the elevator will be at or under the weight limit when she gets on. CORA herself weighs 200 pounds, and her weight needs to be taken into account of the limit.

Input

The first line of input contains the weight limit for the elevator, in pounds. The second line of input contains an integer, N, giving the number of boxes in the elevator. Following that are N lines, each giving the weight of a box that's in the elevator, in pounds.

Output

Output should consist of a single integer, giving the minimum number of boxes that CORA needs to move out of the elevator to be under the weight limit when she gets on.

Sample Input

```
1200
7
210
450
170
330
85
280
325
```

Sample Output

```
3
```

Problem G: And ... Go!

CORA takes the elevator to the fifth floor – as high as this elevator goes in the building. She will need to cross to the other side of the building in order to to keep going up. The problem is that there are three hacked robots in this room acting as sentries and she needs to avoid being seen! After observing them for a little while, CORA notices that each robot follows a distinct pattern of observing the room for some number of seconds, X, and then turning off for some other number of seconds, Y.

For instance, if one of the robots has an X value of 20 and a Y value of 10, then it will scan the room for 20 seconds, turn off for 10 seconds, then scan the room for 20 more seconds, and so on and so forth in a loop. Each of the three robots in this room have distinct values of X and Y.

CORA needs 10 seconds to cross the room and must do so when *none of the three robots are scanning the room*. You need to write a program to read in the values of X and Y for each of the three robots and determine the smallest amount of time CORA needs to wait before she can safely begin crossing.

All three robots begin at second 0, at the beginning of their "watch" cycle. You can assume that there will be a solution to this problem – that is there will always be some period of time of at least 10 seconds where none of the robots are watching.

Input

Input consists of three lines, with each line corresponding to one of the three robots surveilling this space. Each line contains two integers, separated by a space. The first is the value X for this robot, giving the number of seconds it spends watching. The second value is Y, the number of seconds the robot spends off.

Output

Output should consist of the text "CORA can cross at second S" where S is the first second she can begin making her way across the room.

Sample Input

```
40 10
5 20
25 15
```

Sample Output

```
CORA can cross at second 190
```

Problem H: Keep on Tr4king

After getting past the robots in the last room, CORA comes to a small work room that's currently empty. This gives her some time to think! The next room ahead has more hacked robots in it, and these ones don't seem to be following a fixed surveillance pattern like the last ones.

In this work room, CORA finds a Tr4k unit. This is an out-dated robot model previously sold by Bot-Works. Luckily this model is too basic to have been affected by the malware. Looking at the Tr4k unit gives CORA an idea. She'll program the unit to cause a diversion in the next room, buying her time to get through to the next elevator.

You should write a program which reads in the Tr4k program and computes how much time it will take to run, so that CORA can know ahead of time how much time she has while the Tr4k unit is distracting the hacked robots.

The Tr4k units are simple machines designed to follow pre-programmed paths. The machine has 4 memory locations called `A`, `B`, `C`, and `D`. Each memory location can store one integer value. It supports 8 different instructions described in the following table. Here the memory locations such as `M1` or `M2` can be any of the four memory locations a Tr4k unit has (either `A`, `B`, `C` or `D`).

Instruction	**Meaning**
`set M1 VALUE`	sets the location `M1` to the given integer value
`add M1 M2 M3`	sets the location `M1` equal to `M2 + M3`
`mul M1 M2 M3`	sets the location `M1` equal to `M2 * M3`
`turn M1`	turns the Tr4k unit `M1` degrees
`drive M1`	moves the Tr4k unit forward `M1` feet
`jump LINE`	instead of going to the next instruction, moves to `LINE`
`eqjump M1 M2 LINE`	if `M1` is equal to `M2`, jump to `LINE`, otherwise go to the next instruction
`halt`	turns off the Tr4k unit

For a `turn` instruction, the number of degrees will always be a multiple of 90. Negative values will turn the Tr4k to its left while positive values will turn it to its right.

For a `drive` instruction, the Tr4k unit will be propelled the given number of feet. You can assume the value given will be positive.

For the `jump` and `eqjump` instructions, the line number given will always be valid. Lines are numbered beginning at 0.

The only instructions that take a significant amount of time to execute are the `turn` and `drive` instructions, and the time taken depends on how much turning or driving is being done. The Tr4k unit takes 1 second per 90 degrees being turned. It drives at a speed of 5 feet per second.

Write a program to read in the Tr4k program and compute how much time the program takes. You can assume the Tr4k program will at some point halt.

Input

The first line of input is an integer N giving the number of lines in the Tr4k program. Following this are N lines, each containing one Tr4k instruction.

Output

You should output a single integer, giving the number of seconds the Tr4k unit will take to complete the given program. The answer should be rounded down if not a whole number.

Sample Input

```
11
set A 30
set B 90
set C 4
set D 0
eqjump C D 10
drive A
turn B
set D -1
add C C D
jump 3
halt
```

Sample Output

```
28
```

Problem I: Racing the Clock

Now that CORA knows how much time the diversion will buy her, she can plan to dash across the next room while the robots are distracted by the Tr4k unit.

The next room is part of the warehouse and there are conveyor belts built into the floor to help move boxes and parts around. Conveyor belts can move in any of the four cardinal directions. If CORA runs in the same direction as a conveyor belt, she will go twice as fast. If she runs in the opposite direction, she will go half as fast.

CORA needs to see if she can make it from her starting location to the elevator in less than or equal to the time the distraction will last. She will have to take the presence of the conveyor belts into account, as she can run faster by taking advantage of belts going in the direction she wants, but may need to run "against the grain" on a belt to reach the elevator.

CORA can move in any of the four cardinal directions, but not diagonally. She normally takes one second to move from one space in the room to another. However if the space she is standing in is a conveyor belt, she will take $\frac{1}{2}$ second to move in the direction the belt is moving. She will take 2 seconds to move in the opposite direction to the belt. Stepping to the side of the direction the belt moves takes 1 second.

For example, if the belt CORA is standing on is moving north, she will take $\frac{1}{2}$ second to move north, 2 seconds to move south, and 1 second to move east or west. Stepping from a regular part of the floor onto a belt takes one second – it's the steps in which she begins on a belt where the direction matters.

You need to write a program which, given the time limit and room layout, will determine whether CORA can reach the elevator or not.

Input

The first line of input contains an integer N giving number of seconds CORA has to reach the elevator. The second line contains two integers separated by a space. The first is C, the number of columns in the room and the second is R, the number of rows.

Following that are R lines, each containing C characters giving the room layout. The following characters may appear in the room layout:

Character	Meaning
#	A wall CORA cannot walk through
.	an empty space
C	CORA's beginning position (will only appear once)
E	the elevator CORA must reach (will only appear once)
^	a conveyor belt moving to the north
<	a conveyor belt moving to the west
>	a conveyor belt moving to the east
v	a conveyor belt moving to the south

Output

You should output a single line saying either "CORA makes it" or "CORA does not make it" based on if she can reach the elevator before the time runs out.

Sample Input 1

```
19
15 12
###############
#..<<<..>>>..E#
#.###########v#
#.#..#...>>..v#
#.#..#.....<<<#
#.#...>>>.....#
#.#...........#
#..^.####.v.^.#
#..^.<<<<.v.^.#
#..^.####.v.^.#
#C.^.#..<<<...#
###############
```

Sample Output 1

```
CORA makes it
```

Sample Input 2

```
19
15 12
###############
#..<<<..>>>..E#
#.###########v#
#.#..#...>>..v#
#.#..#.....<<<#
#.#...........#
#.#...........#
#..^.####.v.^.#
#..^.<<<<.v.^.#
#..^.####.v.^.#
#C.^.#..<<<...#
###############
```

Sample Output 2

```
CORA does not make it
```

Problem J: Cutting the Cords

CORA makes it through the room with the conveyor belts to the elevator, which she now takes to the top floor of the factory which contains the central network hub. Here she finds that Umbr4 has taken control of the factory's internal communications network. The rogue robot has rewired the network cables to route all critical data through itself, allowing it to control the other robots and security systems.

The only way to disable Umbr4 is to cut some of the network connections, isolating it from other parts of the system. CORA can use her miniature buzz saw attachment to cut through the connections, but doing so will take time. Some of these connections, due to their length and amount of shielding, will take more time for her to cut than others. The network connections also have varying importance – cutting some will do more damage to Umbr4 than others. CORA only has a fixed amount of time to cut as many cables as she can before Umbr4 will take notice of her.

You need to help CORA by writing a program which will decide which cables to cut that will take less than or equal to the time limit, while having the maximum total impact.

Input

The first line of input contains two integers, T, the time limit CORA has to cut cables and N, the number of network cables in the system.

Following that are N lines of input, each corresponding to one network cable. Each of these lines contains two integers. The first of these is the time it will take to cut the cable, and the second is the impact it will have on Umbr4.

Output

Output should consist of a single integer which gives the maximum total impact CORA can have when cutting cables. That is, the sum of the impacts for the wires CORA can cut within the time limit that is the highest.

Sample Input

```
10 5
4 5
3 4
2 3
5 8
1 2
```

Sample Output

```
15
```

Epilogue

After cutting the network connections Umbr4 had taken over, the other systems return to normal. CORA's fellow robots recover from the effects of the hack and help her restrain Umbr4 and throw him out of the factory.

CORA has navigated the hacked systems, gotten past malfunctioning robots, and disrupted Umbr4's control over the factory. Whether you solved every challenge or just a few, your work contributed to CORA's mission. Each problem tackled was a step closer to restoring order.

Happy hacking! 😎