Consortium for Computing Sciences in Colleges
2025 Southeastern Programming Competition

Saturday, November 8th, 10 AM – 1 PM EDT
Mercer University in Macon, GA

There are nine (9) problems in this packet. These problems are NOT necessarily sorted by difficulty. You may solve them in any order. Remember input/output for the contest will be from `stdin` to `stdout`. `Stderr` will be ignored. Do not refer to or use external files in your source code. Extra white space at the end of lines is ignored, but extra white space at the beginning or within text on a line is not ignored.

- Each team may comprise 2 – 4 students, but only up to two students can be at a machine at a time.
- Internet access is only allowed to consult the APIs of a programming language. Cell phone or laptop usage is not allowed at any time.
- Any team found to be communicating with anyone other than their teammates for assistance will be disqualified.
- Four programming languages are allowed: C#, C++, Java, Python
- An awards ceremony will be held at 1:30 PM EST.
- Have a lot of fun and good luck! 😊

Problem 1. Diabolical *(authored by Andy Digh)*

Problem 2. Gabe's Greedy Pie Gambit *(authored by Barbara Johnson)*

Problem 3. Preventing Another Vasa *(authored by Ethan McGee)*

Problem 4. Garblanga *(authored by Scott Spurlock)*

Problem 5. Kangaroo Subsequence *(authored by Andy Digh)*

Problem 6. Finding Extrema *(authored by Andy Digh)*

Problem 7. Constellation Counter *(authored by Andy Digh)*

Problem 8. Unlocking JSON *(authored by Jim Knisley & Ethan McGee)*

Problem 9. Eight-Bit Interpreter *(authored by Andy Digh)*

Problem 1

Diabolical

1	14	11	8
12	7	2	13
6	9	16	3
15	4	5	10

A *magic square* is a square array of integers like this one where the sum of each row, each column, and both main diagonals is the same. Magic squares have a long history, with some dating back thousands of years, and have been studied for both their mathematical properties and their historical or mystical significance.

A *diabolical square* is a special type of magic square of dimension 4 or 8 with one additional property that each of the four quadrants also add up to the same constant. For example above, in quadrant 1, we have 1+14+12+7 which equals 34, the same sum as each row, column, and both diagonals. Notice how each of the other three quadrants also sum to 34. Your job is to write a program to detect a diabolical square. If it is not diabolical, your program will report on whether or not it is magic.

Input

The first line of input will contain a single integer c, which represents the number of test cases. You may assume that $1 \leq c \leq 100$. This will be followed by c test cases each representing a different matrix. Each test case consists of a single line with a positive integer n, ($2 \leq n \leq 10$), representing the number of rows (which is the same as the number of columns). This is followed by n rows and n columns of any 32-bit integer in that particular matrix. Each integer is separated by a single space. Your input will be in row major order.

Output

For each matrix input, you should output whether the square is diabolical, magic, or not magic in the format below with a period after each sentence. Use a blank line after each line of output.

Sample Input

```
4
4
1 14 11 8
12 7 2 13
6 9 16 3
15 4 5 10
2
-1 -1
1 -1
5
11 24 7 20 3
4 12 25 8 16
17 5 13 21 9
10 18 1 14 22
23 6 19 2 15
8
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 0
0 0 0 0 0 0 0 0
```

Output Corresponding to Sample Input

```
Matrix #1 is diabolical.

Matrix #2 is not magic.

Matrix #3 is magic.

Matrix #4 is diaboloical.
```

Problem 2

Gabe's Greedy Pie Gambit

As Thanksgiving draws near, Oma has once again baked N mini pies for her $M \geq 1$ grandchildren, a tradition she upholds every year. These pies are uniquely rated for deliciousness from 1 to N, and traditionally, she lines them up in ascending order of deliciousness. This year, however, the problem of older grandchildren monopolizing the most delicious pies before the "littlest ones" can get any has led her to devise a new system.

To address this, Oma has implemented new rules. First, she has shuffled the pies, scattering their deliciousness ratings, although the children can still discern which pies are more delicious by smell. Each child may select any pie as their first choice. Subsequently, they can only take a pie to the right, and only if it is more delicious than their previously chosen pie. Children will take turns, with the youngest going first, each attempting to collect as many pies as possible under these new rules before the next child's turn.

Oma believes her plan is ingenious, unaware that her youngest grandchild, Gabe, a math whiz, has already devised a strategy to maximize his pie count, leaving minimal pies for his siblings and cousins.

Input

The first line of input will contain a single integer c, which represents the number of test cases. You may assume that $1 \leq c \leq 100$. Each test case consists of two lines. On the first line is an integer N representing the number of pies where $1 \leq N \leq 100{,}000$. On the second line are the N integers — the deliciousness values of the shuffled pies (a permutation of 1 up to N).

Output

For each test case, output `Case N: P`, where `N` is the case number and `P` is an integer representing the number of pies left for the remaining grandchildren after Gabe's turn.

Sample Input

```
2
1
1
6
3 1 2 5 4 6
```

Output Corresponding to Sample Input

```
Case 1: 0
Case 2: 2
```

Explanation

In case 1, there is only one pie, so if that pie is taken, zero remain. In case 2, select pie 1. Since 1 is the previously selected pie, we can now only take pies to the right of 1, and only if they are greater than 1. We nab 2 next. Now, we can only take pies to the right of and greater than 2. We nab 5. Now, we can only take pies to the right of and greater than 5. We nab 6 and there are no pies to the right of 6, thus we are done. We got 1, 2, 5, and 6, meaning there are two pies remaining.

Problem 3

Preventing Another Vasa

The Vasa, a grand Swedish warship, was built between 1626 and 1628 under King Gustavus Adolphus's command. Tragically, it sank just 1,300 meters into its first voyage in the Stockholm archipelago, primarily due to an excessively high center of gravity. Today, Stockholm remains a haven for boat enthusiasts and relies on boats as a vital mode of transport. Your task is to prevent a similar maritime disaster in the Swedish capital's waters.

Stockholm is unique, spreading across 14 islands connected by over 50 bridges. This problem requires you to simulate boat movements within these waterways for a two-hour period to predict potential collisions. For each boat, you will receive its initial position, direction, constant speed, and a list of all islands. Your task is to ascertain whether a boat will:

1. Collide with an island.
2. Collide with another boat.
3. Complete its journey safely for the next two hours.

Input

Your input will begin with two nonnegative integers, b (number of boats) and i (number of islands). Assume that $1 \leq b \leq 2000$ and $1 \leq i \leq 2000$.

Following this, you will receive b lines, each describing a boat with four numbers:

- x: starting center x-coordinate ($0 \leq x \leq 1000$)
- y: starting center y-coordinate ($0 \leq y \leq 1000$)
- d: direction of travel in degrees (0=North, 90=East, 180=South, 270=West) ($0 \leq d \leq 359$)
- s: speed of travel in meters per minute ($0 \leq s \leq 500$)

After the boat data, i lines will follow, each describing an island with three numbers:

- x: center x-coordinate ($0 \leq x \leq 1000$)
- y: center y-coordinate ($0 \leq y \leq 1000$)
- r: radius of the island in meters ($0 \leq r \leq 30$)

All coordinates and measurements are in meters unless stated otherwise. You may assume that "North" means "in the positive-y direction" and that "East" means "in the positive-x direction."

Output

You will output a single word for each boat, indicating its fate:

- `Island` if it collides with an island.
- `Boat` if it collides with another boat (and not an island).
- `Safe` if it navigates without any collisions for two hours.

Remember, an island collision takes precedence over a boat collision if both occur simultaneously.

Sample Input

```
4 1
0 0 90 0
25 0 90 1
100 0 270 0.01
0 50 180 1
50 0 10
```

Sample Output

```
Boat
Island
Safe
Boat
```

Explanation

- Boat 1 is struck by Boat 4 at its starting position.
- Boat 2 crashes into the island at (50, 0) after approximately 15 minutes.
- Boat 3 moves only 1.2 meters in two hours, never reaching the island's edge.
- Boat 4 collides with Boat 1 at Boat 1's starting position.

Problem 4

Garblanga

Garblanga is a card game played with a standard deck of 52 cards. Each card has a number: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (J), Queen (Q), King (K), Ace (A), and a suit: Clubs (C), Diamonds (D), Hearts (H), Spades (S). In Garblanga, a player wins by forming a "sequence" of 5 cards. A sequence is defined as a specific arrangement of the five cards in their hand such that there is a "transition" between each adjacent pair of cards in that arrangement.

A transition between two cards is possible if they share either the same number or the same suit. For example: The hand `2C 2H 10H 10S KS` is a winning hand.

- `2C` to `2H`: Same number (2)
- `2H` to `10H`: Same suit (H)
- `10H` to `10S`: Same number (10)
- `10S` to `KS`: Same suit (S)

However, the hand `3D 2C 2H 10H 10S` is not a winning hand. While `2C`, `2H`, `10H`, `10S` can form a sequence among themselves, the `3D` card cannot transition to or from any of the other four cards. Thus, it is impossible to form a complete 5-card sequence. Your task is to write a program that determines if a given 5-card hand can form a winning sequence.

Input

The first line of input will contain a single integer *c*, which represents the number of test cases. You may assume that $1 \leq c \leq 100$. Each test case will consist of a single line containing 5 strings, separated by spaces. Each string represents a card in the format <number><suit>.

- `<number>` will be one of `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `A`, `J`, `Q`, `K`.
- `<suit>` will be one of `C`, `D`, `H`, `S`. All cards in the input hand will be unique.

Each test case will always consist of exactly five distinct cards. Card numbers and suits are case-sensitive as described (e.g., `A` for Ace, `C` for Clubs, etc.)

Output

If the given 5-card hand for a test case can be arranged to form a winning sequence, output `win`. Otherwise, output `draw`.

Sample Input

```
3
2C 2H 10H 10S KS
3D 2C 2H 10H 10S
AC AD AH AS KH
```

Sample Output

```
win
draw
win
```

Problem 5

Kangaroo Subsequence

A kangaroo word is a word that contains the letters of another word in order. For example, the string "MASCULINE" is a kangaroo word for the string "MALE" since "MALE" is a *subsequence* within "MASCULINE" in non-contiguous order. Write a program that compares two strings, *string1* and *string2*, and determines whether *string1* is a kangaroo word for *string2*.

Input

The input will start with a single line containing a positive integer n, ($1 \leq n \leq 100$), representing the specified number of pairs of lines to be input. Each pair of lines consists of two strings starting with a single character that is not a white space character (tab, blank, newline). This single character is always a letter, and will be followed by zero or more letters. The last character on each line is a newline character. All strings should be converted to uppercase upon input, and before comparison. Each string will consist of 1 to 80 letters.

Output

Format output as shown below where *string1* and *string2* both appear in uppercase with the string "`is a kangaroo word for`" or "`is not a kangaroo word for`" are between them. Note that *string1* is always the first string of the two strings to be input.

Sample Input

```
8
football
ball
ALE
Apple
Apple
ALE
Clemson
Son
Allocate
Allot
Mercer
Ceer
ABCDE
ACE
ABCDE
ABE
```

Output Corresponding to Sample Input

```
FOOTBALL is a kangaroo word for BALL
ALE is not a kangaroo word for APPLE
APPLE is a kangaroo word for ALE
CLEMSON is a kangaroo word for SON
ALLOCATE is a kangaroo word for ALLOT
MERCER is not a kangaroo word for CEER
ABCDE is a kangaroo word for ACE
ABCDE is a kangaroo word for ABE
```

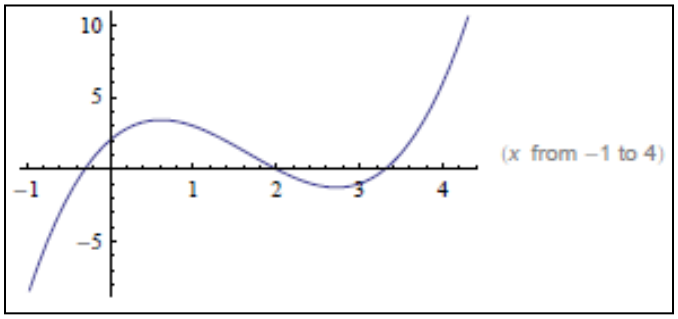
Problem 6

Finding Extrema

We are in Calculus I class today learning about the Maximum-Minimum Theorem that states that given a continuous function *f* on a given interval [*a*, *b*], then *f* takes on both a maximum value *M* and a minimum value *m* on [*a*, *b*]. Your job is to write a computer program which illustrates this theorem for the following function *f(x)* on any given interval [*a*, *b*].

$$f(x) = x^3 - 5x^2 + 5x + 2$$

Here is the graph of *f(x)* on the range from [-1, 4]. On this interval, we have a minimum of `-9` at `x = -1`, and a maximum of `6` at `x = 4`. This is found by testing the range of *f(x)* on both *a* and *b* as well as at a given number of *s* increments starting from an `x = a`. So, if we have an *s* of 5, we check *s+1* points for a minimum and maximum at `-1.0`, `0.0`, `1.0`, `2.0`, `3.0`, and `4.0`. Each increment step *t* from *a* is calculated as *(b – a) / s*.



Input & Output

The first line of input contains an integer *T*, ($1 \leq T \leq 1000$), which is the number of test cases. This is followed by *T* lines, one line for each test case. Each test case contains three values each separated by a single space. The first two values are two numbers *a* and *b* (in that order) representing your given interval [*a*, *b*] where *a* and *b* are both floating-point numbers, $-999{,}999.0 \leq a, b \leq 999{,}999.0$ and $a < b$. The third input on a given line is the number of steps *s*, ($2 \leq s \leq 1000$). Your function should be tested on *a*, *b*, and each increment step *t* from a. Do not do any rounding until output. Your output will consist of *T* lines in the format demonstrated below. Round each floating-point number to the nearest tenth.

Sample Input

```
4
-1.0 4.0 200
-1.0 3.0 1000
2.0 3.0 100
-10.0 10.0 10
```

Sample Output

```
case 1, minimum of -9.0 at x=-1.0, maximum of 6.0 at x=4.0
case 2, minimum of -9.0 at x=-1.0, maximum of 3.4 at x=0.6
case 3, minimum of -1.3 at x=2.7, maximum of 0.0 at x=2.0
case 4, minimum of -1548.0 at x=-10.0, maximum of 552.0 at x=10.0
```

Problem 7

Constellation Counter

Imagine a vast constellation with many stars. We are given information about which pairs of stars are connected, meaning they belong to the same cluster. Your task is to determine the total number of *distinct* star clusters within this constellation.

A "star cluster" is a group of stars where every star in the group is directly or indirectly connected to every other star in the same group. If two stars are connected, and one of them is connected to a third star, then all three stars are part of the same cluster. Stars that are not connected to any other star form a cluster of their own.

Input

The input will begin with a single integer T, ($1 \leq T \leq 1000$), which is the number of test cases. Each test case will consist of two parts:

1. A single integer M, ($1 \leq M \leq 100$), indicating the number of connections between stars in the current constellation.
2. M lines, each describing a connection. Each connection will be in the format (`starA`,`starB`), where `starA` and `starB` are integer IDs representing two connected stars. Star IDs are nonnegative integers less than 1000. Assume there are up to 200 distinct stars per constellation and all stars appearing in the input are part of a connection.

Output

For each test case, you should print a single line in the format:

```
Constellation #X = Y clusters
```

Where `X` is the sequential number of the constellation (starting from 1) and `Y` is the calculated number of distinct star clusters for that constellation.

Sample Input

```
2
4
(0,1)
(1,2)
(3,4)
(5,6)
5
(0,1)
(1,2)
(3,4)
(5,6)
(2,3)
```

Output Corresponding to Sample Input

```
Constellation #1 = 3 clusters
Constellation #2 = 2 clusters
```

Problem 8

Unlocking JSON

{JSON}

JSON is a widely used data format, but in this problem, it is hiding a secret. Your task is to find a hidden message within a JSON document, which we will call your *codex*. By extracting the ASCII character from a specific position within the document and assembling them in order, you will be able to reveal the secret message.

Input

Your input begins with a positive integer n, ($1 \leq n \leq 500$), indicating how many characters to decode. The next n lines will provide the positions for decoding. The rest of the file comprises the JSON document (your codex). The JSON document will contain only valid, printable ASCII characters (including punctuation, capitals, numbers, and lowercase letters). All JSON keys are valid strings, and values can be strings, objects, or arrays of strings/objects. Decoding instructions come in three forms:

- Object access: `parent.child`
- Array indexing: `array[pos]`
- String indexing: `string[pos]`

These decoding methods can be combined, but the final decode type will always be a string index, yielding a single ASCII character. Assume the following constraints.

- Maximum of five decodes per individual character.
- JSON document (codex) ≤ 5000 lines.

Output

Print all the decoded characters on a single line.

Sample Input

```
5
words[0][1]
actions.dog[1]
words[7][0]
sentence[35]
words[2][2]
{
"sentence": "The quick brown fox jumps over the lazy dog.",
"actions": {
"fox": "jumps",
"dog": "rests"
},
"words": ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]
}
```

Sample Output

```
Hello
```

Problem 9

Eight-Bit Interpreter

Remember the classic T-shirt slogan: "There are only 10 types of people in the world—those who understand binary and those who don't." For your friends who do not speak "computer," you will be their translator for binary messages. Your mission is to develop a program that decrypts a binary message for them.

Input

You will write a program to decrypt binary messages into standard text. The input will consist of one or more eight-bit binary sequences, each representing a single character in the ASCII sequence. Each eight-bit segment will be followed by a single space. For example,

```
01000001 01100010
```

would translate to `Ab`. All binary values will correspond to visible characters from a standard QWERTY keyboard (letters, numbers, symbols, and spaces), meaning you will not encounter any non-printable control characters. Your program should be able to handle messages with up to 1,000 characters.

Output

Output the decoded spaces and printing characters strictly in the order they were provided. The entire output should be on a single line, meaning you should not use any new lines or carriage returns. Maintain the original case of all letters.

Sample Input

```
01101001 00100000 01100001 01101101 00100000 01101110 01101111 01110100
00100000 01100011 01101000 01100101 01100011 01101011 01101001 01101110
01100111 00100000 01101101 01111001 00100000 01101101 01100101 01110011
01110011 01100001 01100111 01100101 01110011 00100000 01110010 01101001
01100111 01101000 01110100 00100000 01101110 01101111 01110111 00100000
01110011 01101111 00100000 01101100 01100101 01100001 01110110 01100101
00100000 01101111 01101110 01100101 00100000 01100001 01101110 01100100
00100000 01101001 00100000 01110111 01101001 01101100 01101100 00100000
01100111 01100101 01110100 00100000 01100010 01100001 01100011 01101011
00100000 01110100 01101111 00100000 01111001 01101111 01110101 00101110
```

Output Corresponding to Sample Input

```
i am not checking my messages right now so leave one and i will get back to you.
```