2025 Denison Spring Programming Contest
Granville, Ohio
22 February, 2025

<u>Rules:</u>

1. There are **six** problems to be completed in **four hours**.
2. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
3. No whitespace should appear in the output except between printed fields.
4. All whitespace, either in input or output, will consist of exactly one blank character.
5. The allowed programming languages are C, C++, Python 3, and Java.
6. All programs will be re-compiled prior to testing with the judges' data.
7. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
8. The input to all problems will consist of multiple test cases.
9. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
10. All communication with the judges will be handled by the PC^2 environment.
11. Judges' decisions are to be considered final. No cheating will be tolerated.

Problem A: RailTrail Cycling

The C&O (Chesapeake and Ohio) together with the GAP (Great Allegheny Passage) form a 334 mile gravel trail extending from Washington DC to Pittsburgh, PA for hiking, cycling and eco-tourism. This pathway has mile markers at each mile, starting with Mile 0 in Washington and ending with Mile 334 in downtown Pittsburgh.

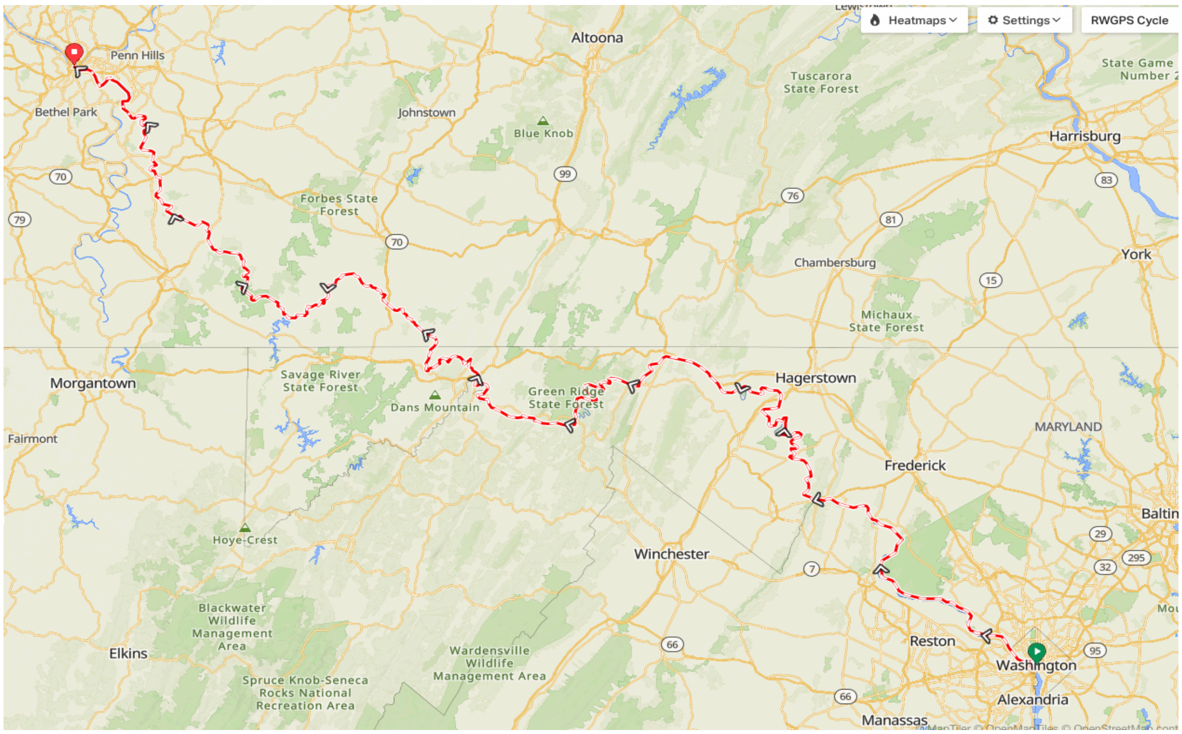


Figure 1: C&O and GAP Trail Map

Veronica lives in this area and regularly completes bicycle trips on the pathway. On each ride, she picks a mile marker to start a ride and picks another mile marker as the turn-around point, then rides back to the staring point. She records each trip in a logbook as a starting mile marker and a turn-around mile marker.

Veronica's friend Michelle knows a few landmarks on the trail. She asks Veronica, "Did you see the statue at Mile 39 on the trail?". Veronica consults her bicycle logbook and sees that she has ridden past this statue four times. On two of her trips she went by it one way, and then again when coming back to other way for a total of four times past this landmark.

The basic problem consists first of the logbook of Veronica's rides on this trail system. Then we get a series of inquiries from Michelle about landmarks at different locations on the trail system. For each landmark, we want to determine how many times Veronica has seen this landmark. If during a single trip, Veronica rides past it, circles back, and then rides past it again, we count this as two visits. If that landmark coincides with Veronica's start location, we count this as two visits (starting out, and then on the return). However, if the landmark coincides with Veronica's turn-around point, we count this as only one visit.

Input

The input data has two sections. The first section begins with integer n (with $1 \leq n \leq 100$) indicating the n entries in Veronica's logbook. Following this are n lines of input, each of which contains two integers a and b ($0 \leq a, b \leq 334$) indicating the starting mile marker of a ride and the turn-around mile marker of a ride. You can assume each of Veronica's rides is at least 2 miles in length ($a \neq b$).

The second section of input indicates Michelle's landmark queries. It begins with integer k (with $1 \leq k \leq 20$) and is followed by k lines of input, each of which is an integer mile marker along the trail (an integer between 0 and 334 inclusive).

Output

For each landmark in the second part of the input data, you are to print out a case number and then the number of times Veronica has ridden past this landmark according to her logbook. Follow the example output formatting below exactly.

Sample Input

```
5
0 102
50 144
250 203
261 75
50 219
4
80
50
310
144
```

Sample Output

```
Case 1: 8
Case 2: 6
Case 3: 0
Case 4: 5
```

Problem B: Favorite States

A new app asks users to rank their top five favorite US states in order. Two people are "friends" on the app if they share at least one state in the same ranking position. For example, Alice, Bob and Carol might each rank their top five as

Rank	1	2	3	4	5
Alice	OH	PA	IN	KY	MI
Bob	FL	GA	IN	VT	ME
Carol	PA	ID	CT	ME	VT

so that Alice and Bob are friends because they both rank Indiana as their third favorite state, while Carol is not friends with either Alice or Bob; though Carol shares some common states in her top five with Alice and Bob, none of them is in the same rank place.

We desire to determine the largest number of app users who can be assembled into a single group such that each pair of people in the group are friends.

Input

Input is given by multiple problem instances. Each problem instance begins with integer $1 \leq n \leq 20$ which indicates the number of users in the app. Each problem instance then has n lines of input where each line has a user name (no spaces), and then five state abbreviations. All input on each line is separated by whitespace. All state abbreviations will be the official two capital letter designations used by the US Postal service.

A value of $n = 0$ indicates the end of input.

Output

For each problem, the user must output a case number and then the size of the largest group of mutual friends that can be assembled.

Follow the example output formatting below exactly.

Sample Input

```
2
Alice  OH  PA  IN  KY  MI
Carol  AK  ID  CT  ME  VT
5
Alice  AL  AK  AZ  AR  CA
Bob    CA  AR  AZ  AK  AL
Carol  CA  CO  CT  DE  FL
Dave   AL  AR  GA  HI  ID
Eve    CA  WY  WA  VT  NH
7
Alice  OH  PA  IN  KY  MI
Bob    FL  GA  IN  VT  ME
Carol  OH  CA  MT  CO  NY
Dave   OH  WY  ID  WA  MA
Eve    LA  PA  MT  WA  TX
```

```
Fred   NW  AZ  NV  WA  ME
Gwen   LA  WY  IN  CO  DE
0
```

Sample Output

```
Case 1: 1
Case 2: 3
Case 3: 5
```

Problem C: Folding Tape Measure

A cloth tape measure is folded in a snake-like pattern. First the end of the tape measure marked 0 inches is placed on the table. The tape measure is stretched 1 inch to the right and then folded back to the left. The tape measure is then stretched 2 inches to the left and again folded back. Then it is stretched 3 inches to the right and folded back again. Each time the tape measure extends one inch further past the last fold in that direction until the tape measure ends.

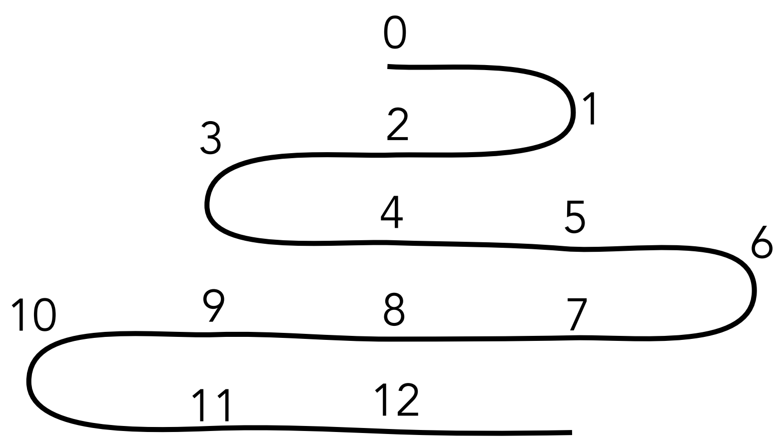


Figure 2: Folded Ruler

Notice that as a result of this folding operation, the marks for Inch 1 and Inch 5 are right on top of each other. The *lateral distance* between them is now 0 inches (assuming infinitesimal distance from the thickness of the ruler). Similarly, the lateral distance between Inch 6 and Inch 12 is now two inches.

For this problem, you will assume you are given a tape measure of length 1000 inches that is folded in the way described. You will then be given n different inch mark pairs (a, b) and you will have to determine the lateral distance between each pair.

Input

The input begins with integer $1 \le n \le 30$. Following are n problem instances, each given by a pair of integers a and b on one line of input. Both $0 \le a, b \le 1000$ are inch marks on the tape measure.

Output

For each problem, you are to print the case number and then the resulting lateral distance between a and b in the folded tape measure.

Sample Input

```
3
0 1
0 2
9 13
```

Sample Output

```
Case 1: 1
Case 2: 0
Case 3: 2
```

Problem D: Calculus Practice

Marta and Julia study calculus together the night before a big quiz. As study practice, their instructor assigns example problems from the book. Marta and Julia have devised a game they play to add an element of fun to their study session. They take turns solving problems. During their turn, they can solve a specific number of problems where the number of problems is chosen from a move set.

For example, tonight the move set consists of $(1, 2, 3)$ meaning that on any given turn, a player may solve 1 problem, 2 problems or 3 problems. If the instructor has assigned Problems 1 through 10 from Chapter 5, then on Marta's turn she may opt to solve problem 5 (one problem), or she may solve problems 5 and 7 (two problems), or she may solve 1, 5 and 8 (three problems). She doesn't have to solve these specific problem; she can pick any one, two or three unsolved problems.

Marta always goes first in the game. They alternate turns until the problems are all solved, or until there are no more legal moves (see below). The person who solves the last problem is declared the winner; the loser must pay for the midnight snack at the campus food court.

To mix things up, each night they select a different move set consisting of three legal moves indicating the exact number of problems that a player may solve during one turn. For example, if the move set is $(1, 4, 5)$ then a player may either solve 1 problem, 4 problems or 5 problems during a turn. In this calculus game, there will always be three options in the move set. Notice that it is possible for the game to end without completing all the problems; for example, if the smallest legal move in the set is 3 and there are 2 problems remaining, then it is not possible to solve these last two problems and the game has ended.

You are given a number of book problems and the three moves in the move set. With the assumption that Marta goes first and they each play optimally, indicate who wins the game. If it is not possible to determine a winner, print "Dutch" indicating each student pays for their own snack.

Input

Each problem instance is given by two lines of numbers. On the first line is an integer $1 \leq n \leq 100$ where n is the number of book problems to solve that night. On the next line are given three integers which indicate the move set (the possible moves on any given turn). A value of $n = 0$ indicates the end of input.

Output

For each problem instance, you are to print the case number and then print the winner of the contest. Print either "Marta" or "Julia" if they are the winner or "Dutch" if no winner can be determined from the given input. Follow the example output formatting below exactly.

Sample Input

```
1
1 2 3
4
1 2 3
15
1 3 8
0
```

Sample Output

```
Case 1: Marta
Case 2: Julia
Case 3: Julia
```

Problem E: Painting Mazes

King Minos of Crete has just created the Bureau of Labyrinth Affairs (BLA) and has appointed you director of the bureau. Your first job is to maintain the elaborate maze that is home to the legendary Minotaur creature. This year you must paint all the interior walls of the labyrinth. To do so, you must figure out how much paint to order.

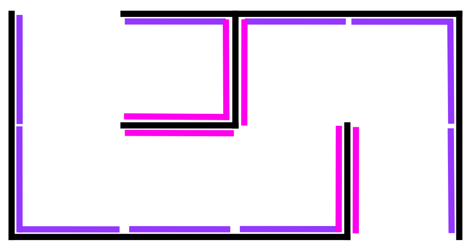
King Minos has two favorite colors: pink and purple. He wants all the interior walls of the maze painted. If both sides of the wall are interior, then paint both faces of the wall pink. If one side of a wall is interior while the other is exterior, then paint the interior side purple. Leave the exterior side unpainted (it has vinyl siding and is good for another 10 years if you power wash it regularly).

The ceiling height of the maze is 20 feet (the minotaur is very big). Each "room" in the maze measures 15 feet wide by 15 feet long. At each point in the maze, there can be a passage UP, DOWN, RIGHT, and/or LEFT. If there is a passage between two rooms, then there is no interior wall (and hence no paint).

If a bucket of paint can cover 1350 square feet of wall space, you must figure out how many buckets of purple and pink paint to order. And then you must hire some painters who are willing to go into the labyrinth and face the minotaur ... but that is another problem.

For example, consider the maze shown below.

(a) Minotaur



(b) Simple Maze

Figure 3: Minotaur and Example Maze

This maze has 8 rooms with 10 exterior wall panels that need to be painted purple on the inside and 6 interior wall panels that need to be painted pink (3 walls, paint each side). You compute the square footage of each panel and determine that 2 buckets of pink paint and 3 buckets of purple paint should suffice.

Input

Each problem instance is a maze and is started with a pair of integers n and m where the maze has n rows and m columns of rooms. Always $1 \leq n, m \leq 10$. Following this are n rows of numbers, each of which contains m values. The rooms are given in row-column order where the first row of integers is the first row in the maze, and the integer values in each row represent the rooms as viewed from left to right.

Each room integer, in the range of 0 to 15, is a 4-bit binary value that indicates the presence of passages in that room. The left-most bit is 1 if there is a passage in this room UP, the next bit is 1 if there is a passage DOWN, the next bit is 1 if there is a passage RIGHT, and finally the right-most bit is 1 if there is a passage LEFT.

The order of the integers is in row-column order (top row, going across left to right, then next row, and so on to the bottom row). You can assume the maze input is consistent; for example, if there is a room with a passage to the RIGHT, then the room immediately adjacent to the right will have a passage to the LEFT.

A value of $n = 0$ and $m = 0$ indicates the end of input.

For example, if a room has the value 14, then this implies

```
        binary                  decimal
   1     1      1      0   =       14
   UP   DOWN  RIGHT  LEFT
```

passages for this room UP, DOWN and RIGHT (but not LEFT). This room is the first room given in the example above and also in the sample data shown below.

Output

For each problem instance, you must figure out first the amount of square footage of all the purple and pink walls, and then the number of buckets of purple and pink paint to order. For each instance, print the case number then integer p which is the number of buckets of purple paint needed, a space, and then integer k which is the number of buckets of pink paint required. Follow the output format demonstrated in the sample data below.

Sample Input

```
2 4
14 1 6 5
10 3 9 12
1 3
3 3 3
0 0
```

Sample Output

```
Case 1: 3 2
Case 2: 2 0
```

Problem F: Random Walk Circumference

Joey takes a random walk on a 2D integer grid. He always starts at the origin, $(0, 0)$, and for each step he goes up, down, left or right one unit. At the end of his random walk with k steps, he has a series of $k + 1$ points of the form (x, y) where x, y are integers. Joey maps his walk by inserting push pins into a piece of graph paper at each point on the walk. He then takes a piece of string and wraps the string around the outside of the collection of pins. Joey wants to compute the distance of the string that encompasses all pins for his walk.

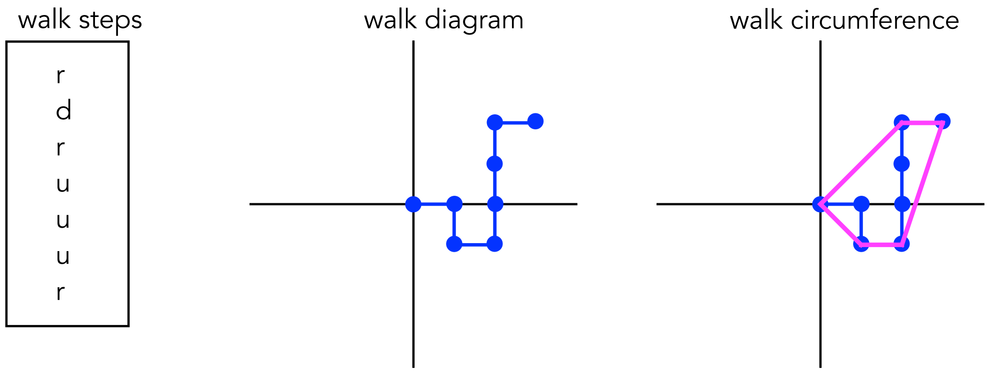


Figure 4: Walk Example

Input

Input is given by multiple problem instances. Each problem instance begins with integer $1 \leq n \leq 100$ which indicates the number of steps in the walk. This is followed by n lines of input where each line contains a character of (u,d,l,r) indicating a movement of up, down, left or right. A problem instance where $n = 0$ indicates the end of input.

Output

For each problem, the user must output a case number and then the circumference of the walk. Since the circumference is a real number, responses must be within ± 0.001 of the actual distance to be judged as correct.

Follow the example output formatting below exactly.

Sample Input

```
3
r
u
l
7
r
d
r
u
u
u
```

```
r
0
```

Sample Output

```
Case 1: 4.0000
Case 2: 9.4049
```