UNIVERSITY OF MANITOBA

DEPARTMENT OF COMPUTER SCIENCE

Manitoba High School Programming Contest 2016

27 May 2016 12:45 – 3:20

Contest Rules:

- Do not open this package until instructed to do so.
- All solutions must be entered completely during the contest. No electronic copies of pre-written code are permitted.
- You may submit as many solutions as you like to each problem; however, incorrect solutions will be assessed a time penalty.
- Contest score is based on the most problems successfully solved; ties are won by shortest total time taken, including any penalties.
- A correct submission must solve the given problem and produce correct output for the given test data within a reasonable time.
- Programming style will not be considered during judging.
- Any programming language resources and notes are allowed.
- No other Internet access is allowed during the contest.

Submission Requirements / Pre-submission Checklist:

A. All input must be read from standard input (in Java, `System.in`). Only Problem 1 does not require input. **Do not open input files.**

B. All output must be written to standard output (in Java, use `System.out.print` or `println`).

C. Your output format must follow the problem requirements **exactly**.

D. Submit the source code file (`.java`, `.c`, etc.–**NOT** `.class`, `.exe`, etc.).

E. Java programs must be in a single file and not placed in a package (no `package` statements; `import` statements are of course OK).

F. Java programs must be complete; **they require a `main` method**.

Problem 1 – Magic 80

The “Magic 80” calculation is used by some retirement plans to determine when employees are allowed to receive retirement benefits. The Magic 80 year for an employee is the year when his or her current age plus the number of years the employee has worked is equal to (or above) 80.

For instance, if an employee starts working at 40, after 15 years of service their age is 55 and the sum is 70, which is not enough to meet the “magic 80” year. However, after 5 more years of employment, the employee is 60 and has 20 years of service (sum is 80), and the employee can retire and receive their retirement benefits. Therefore, their Magic 80 age is 60.

You are in charge of new employees in a company. They have just been hired (have zero years of service) and want to know at what age they will reach their “Magic 80” year, assuming they don’t stop working. You are responsible for printing out a chart that shows employees what their age will be when they reach the “Magic 80” year. If the employee is already at their “Magic 80” year or beyond, the chart shows their current age.

Input

There is no input for this problem. You only need to produce the “Magic 80” output.

Output

You are responsible for printing the chart for all ages from 18 to 81.

For each age, output “If you are YY, you can retire at age XX” where YY is an age and XX is the minimum age when the employee reaches the magic 80 number.

Sample Output

```
If you are 18 you can retire at age 49
If you are 19 you can retire at age 50
If you are 20 you can retire at age 50
If you are 21 you can retire at age 51
```

(ages 22-79 deleted)

```
If you are 80 you can retire at age 80
If you are 81 you can retire at age 81
```

Problem 2 – Collatz Conjecture

The Collatz conjecture is an old unsolved problem in mathematics. The problem has two rules for transforming an integer x:

1. If x is odd, replace x with *3x+1*.
2. If x is even, replace x with **x/2**.

The Collatz conjecture says that no matter what integer x we start with, if we keep applying the two rules, we eventually get to the number 1.

For instance, if we take x = 7, then we get the following steps: 7 → **22** → *11* → **34** → *17* → **52** → **26** → *13* → **40** → **20** → **10** → *5* → **16** → **8** → **4** → **2** → *1*
Thus, the conjecture is true for x = 7. It is not known whether the conjecture is true for all positive integers.

Write a program that counts the number of odd and even steps that it takes for a number to get to 1 by using the Collatz rules. For 7, the number of odd steps is 5 and the number of even steps is 11. For 2, the number of even steps is 1 and the number of odd steps is zero. That is, the number of steps (odd or even) is the number of times the rule is applied while moving to the number 1.

Input

The input consists of an integer N on the first line that gives the number of test cases. Each of the next N lines has a single integer on it, representing the number to test.

Output

For each test case, output the number of odd and even steps needed to get to 1. Write the result as “A. The number XX needs YY odd steps and ZZ even steps.” where A is the case number (starting from 1), XX is the input integer, and YY and ZZ are the number of odd and even steps, respectively. Don’t worry about grammar: “1 even steps” should be printed. There should be a single space after the period in each line of output.

Sample Input	Sample Output

3
7
2
17
	`1. The number 7 needs 5 odd steps and 11 even steps.`
	`2. The number 2 needs 0 odd steps and 1 even steps.`
	`3. The number 17 needs 3 odd steps and 9 even steps.`

Judging Data for Problem 2

```
21
2
4
8
16
32
64
5
21
7
22
11
34
17
176
88
27
54
73
97
129
171
```

Problem 3 – Pivoting

One part of doing a sorting algorithm called “quicksort” is to *partition* the data. Given an unsorted list of integers and a special integer value (called the “pivot”), break the list into two smaller lists: those smaller than (or equal to) the pivot and those larger than the pivot.

For instance, if the unsorted list is

```
93 41 87 5 92 16 45 77 71 51 50 37 72 68
```

and the pivot is 50, then the two lists would be:

```
41 5 16 45 50 37
93 87 92 77 71 51 72 68
```

Note that the elements in the two lists are in the same order that they appear in the original list.

Write a program that takes a list of numbers and a pivot value and produces two lists of integers: the first list are the values that are less than or equal to the pivot, and the second is those greater than the pivot.

Input

The first line of input gives the number N of test cases. The next 2N lines are the N cases: each case consists of two lines.

The first line of each test case has two integers: first the number of integers in the list, and second the pivot value. On the second line of the case are the integers in the list. There may be duplicate values in the list.

Output

For each case, output three lines. The first line should read “-CASE X-“ where X is the number of the case, starting at 1. The second line is the list of values that are less than or equal to the pivot, in the same order as the input. The third line is the list of values that are greater than the pivot, in the same order as the input. If there are no values in a list (if no values are less or greater than the pivot) then leave that line blank in the output.

Sample Input	Sample Output

2
4 100
0 5 100 200
14 50
93 41 87 5 92 16 45 77 71 51 50 37 72 68
	-CASE 1-
	0 5 100
	200
	-CASE 2-
	41 5 16 45 50 37
	93 87 92 77 71 51 72 68

Judging Data for Problem 3

```
9
4 100
0 5 100 200
13 50
93 41 87 5 92 16 45 77 71 51 37 72 68
4 100
0 1 2 3
4 100
101 201 301 401
2 99
100 99
2 99
99 100
3 100
100 99 100
15 5000
812 1823 2533 3636 3639 4351 5261 68 74 82 8692 981031 4112 113114 114
103 75
16 19 26 28 28 37 43 43 44 46 51 59 59 62 66 68 77 80 84 85 89 98 102
104 109 117 16 17 20 28 36 40 42 50 55 55 60 62 62 62 62 66 69 69 74 78
84 91 96 105 105 114 13 14 14 22 22 24 30 30 32 35 35 36 41 47 48 56 59
66 74 74 83 90 90 92 94 96 24 29 38 47 56 62 71 76 83 85 92 98 100 106
108 112 112 115 119 123 126 127 136 141 143
```

[Note: all previous 103 values are on the same line, not five separate lines.]

Problem 4 – MIRROR

ASCII art are pictures constructed from ASCII (basic printable) characters.

```
..#####......#####..
.#.....#....#.....#.
.#..........#.......
.#...........#####..
.#................#.
.#.....#....#.....#.
..#####......#####..
```

In this question, you are asked to read in an ASCII picture consisting of only periods (.) and number signs (#) and output the mirror image. The mirror image is the image that is flipped horizontally along the middle of the picture. For instance, the mirror image of the above image is

```
..#####......#####..
.#.....#....#.....#.
.......#..........#.
..#####...........#.
.#................#.
.#.....#....#.....#.
..#####......#####..
```

Input

There are several test cases. The first line of input gives the number of test cases as an integer on its own line. Each test case starts with the dimensions of the pictures: the width W first and the height H second. Then the next H lines all contain W characters each, and each character is either a period or a number sign.

Output

For each test case, output the number of the case (starting at one) on its own line. Then output the mirror image of the input image.

Sample Input	Sample Output
2 9 9 ..####### .#......# #.....### #....#... #....#... #....#... #.....### .#......# ..####### 20 7 ..#####......#####.. .#.....#....#.....#. .#..........#....... .#...........#####.. .#................#. .#.....#....#.....#. ..#####......#####..	1 #######.. #......#. ###.....# ...#....# ...#....# ...#....# ###.....# #......#. #######.. 2 ..#####......#####.. .#.....#....#.....#. ....#..........#. ..#####...........#. .#................#. .#.....#....#.....#. ..#####......#####..

Judging Data for Problem 4

```
5
9 9
..#######
.#......#
#.....###
#....#...
#....#...
#....#...
#.....###
.#......#
..#######
20 7
..#####......#####..
.#.....#....#.....#.
.#..........#.......
.#...........#####..
.#................#.
.#.....#....#.....#.
..#####......#####..
41 7
.######..#.....#.#.......#######..#####..
.#.....#.#.....#.#.......#.......#.....#.
.#.....#.#.....#.#.......#.......#.......
.######..#.....#.#.......#####....#####..
.#...#...#.....#.#.......#.............#.
.#....#..#.....#.#.......#.......#.....#.
.#.....#..#####..#######.#######..#####..
41 43
.#########.................##########....
....#########................##########..
......#########..............##########..
........#########..........##########....
..........##########.....##########......
............##########.##########........
..............######.##########..........
................##.##########............
.................##########.#............
...............##########.#####..........
.............##########.#########........
...........##########....##########......
.........##########........##########....
.......##########...........##########...
.....##########.............##########...
...##########...............##########...
```

[data continued on next page]

```
..##########...............##########....
..##########.............##########......
..##########...........##########........
...##########........##########..........
.....###########...##########............
.......#########.##########..............
.........#####.##########................
...........#.##########..................
...........##########.###................
.........##########.#######..............
.......##########..##########............
.....##########......##########..........
...##########..........##########........
..##########.............##########......
..##########...............##########....
..##########.................##########..
...##########.................##########.
.....##########...............##########.
.......##########.............##########.
.........##########..........##########..
...........##########......##########....
.............##########..##########......
...............#######.##########........
.................####.#########..........
....................#########.#..........
..................##########.###.........
................###########.#######......
55 15
#.....#.######...#####...#####..######.....#....#.....#
##...##.#.....#.#.....#.#.....#.#.....#...#.#....#...#.
#.#.#.#.#.....#.#.......#.......#.....#..#...#....#.#..
#..#..#.######..#........#####..#.....#.#.....#....#...
#.....#.#.....#.#.............#.#.....#.#######....#...
#.....#.#.....#.#.....#.#.....#.#.....#.#.....#....#...
#.....#.######...#####...#####..######..#.....#....#...
.......................................................
.............#####....###.....#....#####...............
............#.....#..#...#...##...#.....#..............
..................#.#.....#.#.#...#....................
#####.#####..#####..#.....#...#...######..#####.#####..
............#.......#.....#...#...#.....#..............
............#........#...#....#...#.....#..............
............#######...###...#####..#####...............
```

Problem 5 – Jaccard Index

The Jaccard Index is a measurement of how similar two sets are. If you have two sets X and Y, the formula for computing the Jaccard Index is

$$\frac{|X \cap Y|}{|X \cup Y|}$$

where $|X \cap Y|$ is the number of items that are in **both** X and Y (in the intersection of X and Y) and $|X \cup Y|$ is the number of items that are in **either** X or Y (in the union of X and Y). For instance, if X = {1,2,3,4} and Y = {2,3,4,5} then there are three elements in both X and Y (namely 2,3 and 4) and five elements in either X or Y (namely 1,2,3,4 and 5). So the Jaccard Index is 3/5 = 0.6.

Write a program that computes the Jaccard Index of two sets of integers and prints the result with one decimal place.

Input

There are several test cases. The first line of input gives the number of test cases as an integer on its own line. After the first line, each test case consists of three lines: the first line has two integers s and t, giving the number of elements in each of the two sets. The second line has s integers, which are the elements in the first set, while the third line has t integers, which are the elements in the second set. The elements in the sets are in no particular order.

Output

For each test case, write "X. Jaccard Index: N" where X is the number of the test case (starting at 1) and N is the Jaccard Index of the two sets, given to one decimal place. There is a single space after the period and a single space after the colon.

Sample Input	Sample Output

3
3 2
0 1 2
3 4
3 3
1 2 3
3 2 1
4 4
1 2 3 4
2 3 4 5
	1. Jaccard Index: 0.0
	2. Jaccard Index: 1.0
	3. Jaccard Index: 0.6

Judging Data for Problem 5

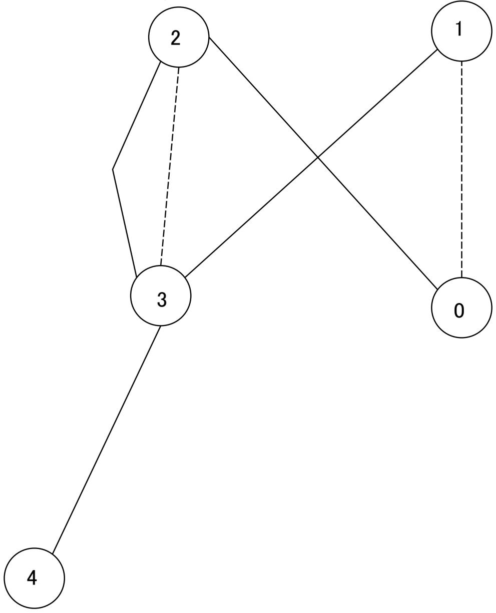
```
8
3 2
0 1 2
3 4
3 3
1 2 3
3 2 1
4 4
1 2 3 4
2 3 4 5
5 4
1 2 3 4 5
2 3 4 5
5 6
2 4 6 8 10
2 3 4 6 8 10
10 1
1 2 3 4 5 6 7 8 9 10
1
10 1
1 2 3 4 5 6 7 8 9 10
20
19 20
62 9 57 64 33 107 66 103 96 100 19 85 60 32 95 42 98 81 40
98 87 95 122 9 53 7 30 1 57 39 54 82 93 75 88 59 89 74 62
```

Problem 6 – Stuck on the Mountain

Ski resorts typically have several chair lifts that take skiers up the mountain, and ski runs that take them down the mountain. Most also have a chalet, where skiers begin and end their day.

A variety of runs is a good idea for a ski resort, but so is ensuring that skiers don't get stuck on the mountain. In this question, you are given a list of runs and lifts, and asked to determine if skiers are able to return to the lodge from any point on the mountain. By returning to the lodge, we mean being able to take a series of lifts and runs so that they end up at the lodge.

For instance, consider the following map of a resort.



In this picture, the lodge is at location 0, the dotted lines are the chair lifts (from 0 to 1 and from 3 to 2) and solid lines are ski runs (the rest). In this resort, locations 0-3 are acceptable, since from each of them we can take a series of lifts and runs to go back to the lodge. For example, from location 3, we would travel from 3 to 2 and then down the ski run to 0. However, location 4 leaves skiers stranded, as they cannot take a lift or run to get back to the lodge.

The ski runs and chair lifts at the resort have been designed so that every run starts at the end of either a lift or another run, and every run ends at the start of a lift or another run. These starting and ending points will be numbered starting at 0, and each lift and run will be given as a list of two numbers: the starting point of the lift/run and the end point of the lift/run. The chalet is always at location number 0. Lifts and runs are both one-way: you aren't allowed to ride lifts down the mountain, and you don't *want* to walk up runs.

Your output should either indicate that all skiers can return to the chalet (“home safe”) or that there are points where a skier can’t get back to the chalet (“stranded”). You don’t need to identify where on the mountain the skiers would be stranded, only that such a place exists.

Input

The input consists of the design for several resorts. The first line of input is a single integer n, giving the number of resorts. Following this are n descriptions of resorts: each starts with two numbers on a line: L and R. L is the number of locations in the resort and R is the number of lifts and runs in the resort. The locations will be numbered 0 to L-1. The next R lines of input in each test case give descriptions of the lifts and runs. Each is a pair of numbers i j, which indicates that there is either a lift or run that goes from location i to location j. The input doesn’t distinguish between runs and lifts: only that you can go from i to j whether it’s uphill or downhill. There is a blank line before each resort.

Output

For each resort, either output “home safe” or “stranded” depending on whether or not you can make it back to the chalet from all places on the mountain. Start each output with a “CASE XX:” where XX is the case number starting with case 1. There is a single space after the colon.

Sample Input	Sample Output

2
 	
4 5
0 1
1 3
2 0
2 3
3 2
 	
5 6
0 1
1 3
2 0
2 3
3 2
3 4
	CASE 1: home safe
	CASE 2: stranded

Judging Data for Problem 6

```
9

2 1
0 1

2 2
0 1
1 0

3 2
0 1
0 2

3 3
0 1
0 2
1 2

3 3
0 1
1 2
2 0

4 5
0 1
1 3
2 0
2 3
3 2

5 6
0 1
1 3
2 0
2 3
3 2
3 4
```

[data continues on next page]

```
14 25
0 1
1 2
2 3
3 0
2 4
4 0
4 5
5 0
5 1
2 6
6 5
8 6
6 9
8 9
9 7
7 8
8 7
8 12
12 13
13 7
13 11
11 10
10 11
10 13
12 7
```

```
14 24
0 1
1 2
2 3
3 0
2 4
4 0
4 5
5 0
5 1
2 6
6 5
8 6
6 9
8 9
9 7
7 8
8 7
8 12
12 13
13 11
11 10
10 11
10 13
12 7
```