Computer Science Competition
Invitational B 2020
Programming Problem Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Aran
Problem 2	Brianna
Problem 3	Deepa
Problem 4	Emilio
Problem 5	Guoliang
Problem 6	Irene
Problem 7	Josey
Problem 8	Kelly
Problem 9	Maciej
Problem 10	Nitin
Problem 11	Teresa
Problem 12	Willie

1. Aran

Program Name: Aran.java **Input File: None**

Aran is beginning the process of teaching his little brother how to write the alphabet. He wants to teach him how to write a new letter, upper and lower case, each day as well as review the letters learned previously. He would like you to write a program that prints the schedule he will need to follow to accomplish this.

There is no input for this problem.

Exact Output:

```
Day 1: Aa
Day 2: AaBb
Day 3: AaBbCc
Day 4: AaBbCcDd
Day 5: AaBbCcDdEe
Day 6: AaBbCcDdEeFf
Day 7: AaBbCcDdEeFfGg
Day 8: AaBbCcDdEeFfGgHh
Day 9: AaBbCcDdEeFfGgHhIi
Day 10: AaBbCcDdEeFfGgHhIiJj
Day 11: AaBbCcDdEeFfGgHhIiJjKk
Day 12: AaBbCcDdEeFfGgHhIiJjKkLl
Day 13: AaBbCcDdEeFfGgHhIiJjKkLlMm
Day 14: AaBbCcDdEeFfGgHhIiJjKkLlMmNn
Day 15: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOo
Day 16: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPp
Day 17: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQq
Day 18: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRr
Day 19: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSs
Day 20: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTt
Day 21: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUu
Day 22: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVv
Day 23: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWw
Day 24: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXx
Day 25: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYy
Day 26: AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz
```

2. Brianna

Program Name: Brianna.java **Input File: brianna.dat**

In Brianna’s statistics course, Brianna just learned that the range of a set of data is the difference between the largest and smallest values. Brianna knows that this is a relatively easy statistic to calculate when the sample set is small, but when the sample set is big, Brianna knows that a careless mistake of missing one value could result in calculating the range incorrectly. In order to prevent careless mistakes when calculating the range, Brianna has decided to write a program that will calculate the range for her. Will you be able to assist Brianna with this?

Input: Input consists of a single list of N positive integers, 5 <= N <= 50, with single space separation, all on one line.

Output: Output the minimum value, maximum value, and the range of the list of integers.

Sample Input:
```
43 25 49 81 5 45 100 81 98 89 95 94 31 14 20 1 61 49 84 73
```

Sample Output:
```
1 100 99
```

3. Deepa

Program Name: Deepa.java **Input File: deepa.dat**

April is closer than you think! Deepa wants to take care of her taxes ahead of time. The United States, like most countries, has a **progressive tax system**, where the tax rate increases with income. To compute how much she has to pay in taxes, Deepa must look up the year's tax brackets.

A tax bracket is a table with dollar values and tax rates for incomes above the dollar values. This is best explained by example. Consider the following simple tax bracket:

$0.00 10%

$9,525.00 12%

$38,700.00 22%

This means that the first $9,525.00 of income will be taxed at 10 percent. Additional income up to $38,700.00 will be taxed at 12%, and any income beyond that is taxed at 22%. Say Deepa's income is $40,000.00. Then the tax she must pay is:

($9,525.00 * 0.10) + ($29,175.00 * 0.12) + ($1,300.00 * 0.22) = $4,739.50

Since she paid $4,739.50 out of her income of $40,000.00, her "effective" tax is approximately 11.849%. Given the tax table for the year and Deepa's income, how much will she owe in taxes. Furthermore, what is her effective tax rate?

Input: The first line of input has an integer T, the number of test cases. Each test case begins with positive integers N and X, the number of tax brackets and Deepa's taxable income for the year. The next N lines each have two integers, the value in the bracket and the amount by which it is taxed. The brackets are given in ascending order of both dollar value and tax amount. The value in the first bracket is always $0.00. There are at most 20 tax brackets. All dollar values are nonnegative and under $1,000,000.00. All tax percentages are expressed as whole numbers under 100.

Output: For each test case, output the amount she must pay in taxes and the effective tax rate. Include a dollar sign and comma separators for the amount she must pay. Always display two decimal places for cents using normal rounding rules. Display the tax rate rounded to exactly 3 decimal places using standard rounding rules. Format the output as in the sample.

Sample input:

```
2
3 40000
0 10
9525 12
38700 22
4 9823
0 10
9525 12
38700 22
82500 24
```

Sample output:

```
Case #1: $4,739.50 11.849%
Case #2: $988.26 10.061%
```

4. Emilio

Program Name: Emilio.java **Input File: emilio.dat**

Emilio has been running almost daily to prepare for spring track. He has been keeping a daily log of the distances he runs and wants to know the average distances for his 10 best and 10 worst days. Emilio is friends with a couple of the UIL programmers and asked your team to help.

Input: First line will contain the number T of test cases with 1 ≤ T ≤ 15. Each test case will start with a line that contains the number N of daily distances that follow (20 ≤ N ≤ 100) which is then followed by an unknown number of lines with unknown number of distances on each line, but still an overall total of N distances for the data set. Emilio has been pretty aggressive with his running but has also had some easy days. Individual distances will be between 3.0 and 10.0 and are rounded to one decimal place. He does not log days on which he did not run.

Output: Each test case will display two lines which start with the case number followed by a colon and a single space. The first line of each test case displays the average of his 10 longest runs and second line displays the average of his 10 shortest runs, both rounded to exactly one decimal place. Each test case will be followed by a line containing 6 dashes “------”.

Sample input:

```
3
25
5.6 3.3 3.5 6.2 4.1 6.3 6.4
6.1 7.6 4.4 3.0 7.8 4.2
4.6 3.4 8.5 4.2 6.6 9.5 9.7 9.3 3.5 7.2
5.5 3.5
35
8.5
5.9 8.5 9.9 9.8 8.3 7.2 6.9
3.8 9.3 6.2 7.7
4.2 7.1 3.4 7.2 4.5 3.2 5.9
5.2 6.8 7.7
9.6 5.5 8.1 10.0
6.2 3.7 5.2 5.8 3.7 4.3 9.1 9.1
3.9
27
7.7 6.2 9.3 9.7 8.5 8.2 8.2 3.4
3.3 9.8 9.2 6.8 3.1 9.7 8.6 7.5
6.1 8.5 5.6 6.1 9.9 5.7 3.7 5.2
4.5 10.0
8.3
```

Sample output:

```
1: 7.9
1: 3.7
------
2: 9.2
2: 4.0
------
3: 9.3
3: 4.7
------
```

5. Guoliang

Program Name: Guoliang.java **Input File: guoliang.dat**

Guoliang would like to try to write a poem using only isograms. An isogram is a word in which no letter of the alphabet occurs more than once. Examples of such words are **go**, **jam**, **backup** and **campgrounds**. He has a long list of words from which to choose but needs to know which ones are isograms. Your team should write a program that will read Guoliang's list of words and print only those that are isograms.

Input: A file containing an unknown number of English words listed on an unknown number of lines each separated by a space. No words contain any symbols, such as contractions or hyphens, just lowercase letters.

Output: Every word from the input file that qualifies as an isogram, each output on a separate line.

Sample input:

```
position tickle tub truculent
erratic chance
equal string bathe
butter need
irate
```

Sample output:

```
tickle
tub
equal
string
bathe
irate
```

6. Irene

Program Name: Irene.java **Input File: irene.dat**

Semiprime numbers are an important part of the algorithms used to develop public and private keys in the field of public key cryptography. Irene has recently taken a job for a security company that develops communications security systems for various clients. Her job is to develop a program that will determine if a value is a semiprime.

A semiprime number is a composite number that is the product of two (possibly equal) prime numbers.

Discrete semiprime numbers (sometimes called **square free semiprimes**) are those that are not squares of prime numbers. Examples include 15, 26 and 35.

3 X 5 = 15

2 X 13 = 26

5 X 7 = 35

49 is semiprime but it is not discrete because it is the product of 7 X 7.

Help out Irene by writing a program that will determine if a number is a **semiprime**, **discrete semiprime** or **neither**.

Input: A number N that represents the number of values to be checked followed by N values X, each on a separate line, where for every X value, 2 < X < 2,147,483,647.

Output: For each number print the value followed by a single space and DISCRETE SEMIPRIME, SEMIPRIME or NOT SEMIPRIME.

Sample input:

```
5
15
16
49
35
72
```

Sample output:

```
15 DISCRETE SEMIPRIME
16 NOT SEMIPRIME
49 SEMIPRIME
35 DISCRETE SEMIPRIME
72 NOT SEMIPRIME
```

7. Josey

Program Name: Josey.java **Input File: josey.dat**

While doing assigned geography reading, Josey came across a surprising discovery: there are only two doubly-landlocked countries on Earth! A country is landlocked if it has no borders with the ocean. A country is doubly-landlocked if all its borders are with landlocked countries. Note that by definition, all doubly-landlocked countries are landlocked themselves. Using the international borders on Earth in late 2019, Josey noted that only Liechtenstein and Uzbekistan are doubly landlocked.

Josey is now wondering which countries are doubly landlocked in the lands of fantasy novels. Since many fantasy novels don't come with globes, Josey has instead written down every occurrence of two neighboring nations in the book, as well as any mention of a nation bordering the ocean. Given this list of borders, can you help Josey find which nations are doubly-landlocked?

Input: The first line of input has a positive integer T, the number of test cases, at most 20. Each test case begins with a single positive integer B, the number of borders Josey has found. B is at most 1,000. The following B lines each contain two space-separated strings, the entities on either side of the border. Entities are either the names of countries or the string "OCEAN", representing the ocean. All country names are at most 20 characters long, and contain only uppercase letters. It is guaranteed at least one country has a border with the ocean.

Output: For each test case, first output K, the number of doubly-landlocked countries. On the subsequent K lines, output the names of the doubly-landlocked countries in lexicographical order. Format the cases as in the samples.

Sample Input:

```
3
13
LIECTENSTEIN SWITZERLAND
SWITZERLAND AUSTRIA
AUSTRIA LIECTENSTEIN
ITALY SWITZERLAND
SWITZERLAND FRANCE
FRANCE ITALY
ITALY OCEAN
OCEAN FRANCE
AUSTRIA SLOVENIA
SLOVENIA OCEAN
GERMANY AUSTRIA
GERMANY SWITZERLAND
OCEAN GERMANY
7
OCEAN ONE
ONE TWO
TWO THREE
THREE FOUR
FOUR FIVE
```

```
FIVE SIX
SIX OCEAN
13
VALE CROWNLANDS
WESTERLAND REACH
REACH STORMLANDS
OCEAN DORNE
STORMLANDS DORNE
RIVERLANDS WESTERLAND
RIVERLANDS NORTH
OCEAN VALE
OCEAN CROWNLANDS
WESTERLAND OCEAN
CROWNLANDS RIVERLANDS
RIVERLANDS OCEAN
REACH OCEAN
```

Sample Output:

```
Case #1: 1
LIECTENSTEIN
Case #2: 2
FOUR
THREE
Case #3: 0
```

8. Kelly

Program Name: Kelly.java **Input File: kelly.dat**

Kelly's great-grandfather was an American prisoner of war (POW) during the Vietnam War. They were often kept isolated from one another and had to be creative to communicate with each other between cells. Talking would result in torture so an alternative approach was necessary. One method widely used during that war was the Tap Code or Knock Code which actually dates all the way back to the ancient Greeks.

The Tap Code uses a two-digit sequence to identify each letter as shown in the following grid with the first tap count as the row number and the second tap count as the column number. The 'C' and 'K' are considered the same character because, in many situations, they contribute the same sound as in "CAT" and "KITE". POWs would tap on the cell walls with a short break between taps and a longer break between words. A tap pattern of 4-taps, 5-taps would be a 'U' followed by 2-taps, 4-taps for an 'I' and then 3-taps, 1-tap for 'L'. I really doubt many POWs were sending messages about UIL but they could! For this programming problem, use taps counts of 0 0 for a space between words.

Tap Codes	1	2	3	4	5
1	A	B	C/K	D	E
2	F	G	H	I	J
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Here is an example:

- Tap count data: `4 5 2 4 3 1 0 0 4 2 3 4 1 3 1 3 4 3`
  interpreted as pairs (4,5) → "U" (2,4) → "I" (3,1) → "L" (0,0) → space
  (4,2) → "R" (3,4) → "O" (1,3) → "C" (1,3) → "C" (4,3) → "S"
  decodes as: "UIL ROCCS"

POWs did not care about punctuation marks and they could spell out digits if needed.

Write a program to decode messages from Kelly to her friends using the Tap Code.

Input: An unknown number of lines [1 ≤ lines ≤ 25] of input with each containing an unknown number of integers, count ≤ 150, each in the range 0 … 5 separated by single spaces. The lines will always contain an even number of counts and no extra characters.

Output: Each line of input will produce one decoded message that is displayed on a single line.

Sample input:
```
4 5 2 4 3 1 0 0 4 2 3 4 1 3 1 3 4 3
4 4 1 1 3 5 0 0 1 3 3 4 1 4 1 5 0 0 5 2 3 4 4 2 1 3 4 3
3 5 4 2 3 4 2 2 4 2 1 1 3 2 3 2 2 4 3 3 2 2 0 0 2 4 4 3 0 0 2 1 4 5 3 3
```

Sample output:
```
UIL ROCCS
TAP CODE WORCS
PROGRAMMING IS FUN
```

9. Maciej

Program Name: maciej.java **Input File: maciej.dat**

Maciej *(pronounced "mah chay")* and his programming partner Mateusz *(pronounced "mah teh oosh")* are designing a website. Both Maciej and Mateusz want to use their favorite color as the main color on the website.

One way to represent colors on a computer is in RGB space. A color is comprised of some amount of red, green, and blue. The amount of each color must be a non-negative integer at most 255. For example (0, 0, 255) is pure blue, (255, 255, 255) is white, and (191, 87, 0) gives a strange orange-brown color.

When displayed, colors usually begin with an octothorpe (this symbol: #), followed by the amounts of each color in hexadecimal, using uppercase letters. (0, 0, 255) becomes #0000FF, and (191, 87, 0) becomes #BF5700.

Maciej and Mateusz decided to compromise in the following manner. They each write down their favorite color. Then, for each color channel, they find a new color that minimizes the maximum difference to each of the original colors and use that. If there are multiple, they choose the smaller one.

For example, if the original colors were #BF5700 and #430030, then the red channel would be 129, the green channel would be 43, and the blue channel would be 24. This would be shown as #812B18.

Input: The first line of input has an integer T, the number of test cases. Each test case has two space-separated colors. All colors will be formatted as explained in the statement using uppercase letters.

Output: For each test case, output the color that the two compromise on, in hexadecimal format, using uppercase letters. Format each case as in the samples.

Sample Input:
```
2
#000000 #FFFFFF
#BF5700 #430030
```

Sample Output:
```
Case #1: #7F7F7F
Case #2: #812B18
```

10. Nitin

Program Name: Nitin.java **Input File: nitin.dat**

Nitin owns and operates the Happy Home For Wayward Pets. HHFWP is a shelter for lost and abandoned cats and dogs. HHFWP operates on a strict protocol for adoption that says the animal that has been at the shelter for the longest time will be the next to be adopted. Those wishing to adopt may choose to adopt a dog or a cat but they may not choose a specific animal. Adopters must take the cat or dog that has been at the shelter the longest. Unfortunately, sometimes an adopter will request a certain species and all that species have already been adopted. These potential pet owners are placed on a wait list and given a call when a new cat or dog comes in. Write a program that will help Nitin print a list of adoptions or add a person on the wait list.

Input: The first line will have a single number P that represents the total number of pets currently up for adoption. The next P lines will list the name and species of the pets at the shelter in the order they arrived at the shelter, where the first pet has been at the shelter for the longest amount of time. The name and species will be separated by a single space. The species will be indicated by a capital D for dogs and a capital C for cats.

The next line following the list of pets will have a single number A that represents the number of people who wish to adopt a pet. The next A lines will list the name of the adopter, in the order they arrived at HHFWP, followed by the species (C or D) each person wishes to adopt.

Output: A list of adoptions or wait list entries each on a separate line. Each line should contain the name of the adopter, the species, and the name of the pet. The species should be spelled out as DOG or CAT. If there is not a pet available in the species requested, the entry should contain the adopters name, WAIT LIST, and the species they have requested each separated by a space.

Sample input:

```
10
Willie D
Ralph D
Rover D
Tiger C
Daisy D
Max C
Felix C
Lucy D
Bubba C
Bella C
10
Noah C
Tereza D
Bruce D
Olivia D
Terry C
Irene D
Joseph D
Estaban C
Rashi D
Emily C
```

Sample output:

```
Noah CAT Tiger
Tereza DOG Willie
Bruce DOG Ralph
Olivia DOG Rover
Terry CAT Max
Irene DOG Daisy
Joseph DOG Lucy
Estaban CAT Felix
Rashi WAIT LIST DOG
Emily CAT Bubba
```

11. Teresa

Program Name: Teresa.java **Input File: teresa.dat**

Teresa's big sister has been learning about matrix operations in her college math class. She doesn't really understand many of the details but has decided to try programming an experimental operation as a personal challenge; however, she is struggling. She is trying to perform a 2-way sort of a matrix so the smallest value is in the top left corner and the largest value is in the bottom right corner. Her experimental operation first sorts within the columns and then across the rows. The result should be that the values increase from left-to-right and top-to-bottom as shown in the following example.

Original Matrix			
73	95	72	15
47	21	41	66
23	61	38	87
98	28	57	39
37	60	22	49

Columns Sorted			
23	21	22	15
37	28	38	39
47	60	41	49
73	61	57	66
98	95	72	87

Final Matrix			
15	21	22	23
28	37	38	39
41	47	49	60
57	61	66	73
72	87	95	98

Can your UIL team help Teresa with this sorting challenge?

Input: First line will contain a number 1 ≤ T ≤ 10 as the number of test cases. The first line of each test case will contain two whole numbers R and C, the number of rows and columns in the matrix, both will be ≥ 2 and ≤ 25. The following R rows will each contain exactly C integers, separated by white space, which is the data for the matrix. All data values will be ≥ 100 and < 1000.

Output: For each test case, output one line containing the test case number followed by a colon. The next R rows display the data in the columns of the sorted matrix as shown below, with exactly one tab following each data value. Below each test case, display a single line containing 12 plus signs, "++++++++++++"

Sample input:
```
2
5 4
194    819    449    405
560    410    914    534
302    670    856    448
933    239    259    477
591    455    665    652
3 5
558    777    773    761    995
252    632    580    639    818
590    828    372    386    489
```

Sample output:
```
1:
194    239    259    405
302    410    448    449
455    477    560    665
534    591    670    856
652    819    914    933
++++++++++++
2:
252    372    386    489    632
558    580    639    777    818
590    761    773    828    995
++++++++++++
```

12. Willie

Program Name: Willie.java Input File: willie.dat

Willie and his buddies in the chess club just stumbled across the N-Queen problem. The N-Queen problem is the problem of placing N chess queens on an N×N chessboard so that no two queens can capture each other. Recall, that in chess a queen can move any number of squares vertically, horizontally or diagonally. Capture is defined as when a Player P1's chess piece moves onto the same square as an opponent P2's piece. When P2's piece is captured, the captured piece is removed from the board and P1's piece must remain in the square until the next turn. An example solution to the 4 Queen problem is below:

```
  C0   C1   C2   C3
    -----------------
R0 |    |    | Q  |    |
    -----------------
R1 | Q  |    |    |    |
    -----------------
R2 |    |    |    | Q  |
    -----------------
R3 |    | Q  |    |    |
    -----------------
```

As can be seen, no two queens would be able to capture each other no matter the movement made. An example of an incorrect attempt at the 4 Queen problem is below:

```
     C0   C1   C2   C3
    -----------------
R0 |    |    | Q  |    |
    -----------------
R1 | Q  |    |    |    |
    -----------------
R2 |    |    |    | Q  |
    -----------------
R3 |    |    | Q  |    |
    -----------------
```

Notice, the queen at (R1,C0) could move diagonally to (R3,C2) thus capturing the queen, or the queen at (R3,C2) cold move diagonally to (R1,C0) thus capturing the other queen. The queen at (R0,C2) could move down to (R3,C2) thus capturing the queen, or the queen at (R3,C2) could move up to (R0,C2) thus capturing the other queen. And finally, the queen at (R2,C3) could move diagonally to (R3,C2) thus capturing the queen, or the queen at (R3,C2) could move diagonally to (R2,C3) thus capturing the other queen. This therefore is NOT a solution to the 4 Queen problem. In this case, there were a total of 3 pairs of queens that could capture each other. For a solution to the N-Queen problem, there must be zero pairs of queens in position to capture each other.

Willie and his buddies have challenged each other to try and solve the N-Queen problem for 1 <= N <= 10. They need your help writing a program that will validate their attempts as either correct solutions to the N-Queen problem, or incorrect solutions to the N-Queen problem. Can you help them with this?

(Continued next page)

(Willie – continued)

Input: Input starts with a line containing an integer X (1 <= X <= 15), the number of test cases. The following X test cases will each consist of two parts: 1) N, and 2) the NxN board with the queens placed. Each test case will be followed by 50 asterisks. This line serves as a separator between each test case.

NOTE It is not given that N queens will be placed.

Output: For each test case output that the attempt is either a "valid solution" or "incorrect attempt" at the N-Queen problem. To be a "valid solution" N queens must be placed on the board and no two queens can capture each other.

Sample input:

```
4
8
---------------------------------
|   |   |   | Q |   |   |   |   |
---------------------------------
|   | Q |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   | Q |   |
---------------------------------
|   |   | Q |   |   |   |   |   |
---------------------------------
|   |   |   |   |   | Q |   |   |
---------------------------------
|   |   |   |   |   |   |   | Q |
---------------------------------
|   |   |   |   | Q |   |   |   |
---------------------------------
| Q |   |   |   |   |   |   |   |
---------------------------------
**************************************************
4
-----------------
|   |   | Q |   |
-----------------
|   | Q |   |   |
-----------------
|   |   |   | Q |
-----------------
|   | Q |   |   |
-----------------
**************************************************
10
-----------------------------------------
|   |   |   |   |   |   |   | Q |   |   |
-----------------------------------------
|   |   |   |   | Q |   |   |   |   |   |
-----------------------------------------
|   | Q |   |   |   |   |   |   |   |   |
-----------------------------------------
|   |   |   |   |   |   |   |   |   | Q |
-----------------------------------------
|   |   |   |   |   |   | Q |   |   |   |
-----------------------------------------
|   |   |   | Q |   |   |   |   |   |   |
-----------------------------------------
| Q |   |   |   |   |   |   |   |   |   |
-----------------------------------------
|   |   |   |   |   |   |   |   | Q |   |
-----------------------------------------
|   |   |   |   |   | Q |   |   |   |   |
-----------------------------------------
|   |   | Q |   |   |   |   |   |   |   |
-----------------------------------------
**************************************************
8
---------------------------------
|   |   |   | Q |   |   |   |   |
---------------------------------
| Q |   |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   |   | Q |   |
---------------------------------
|   | Q |   |   |   |   |   |   |
---------------------------------
|   |   |   |   |   | Q |   |   |
---------------------------------
|   |   |   |   |   |   |   | Q |
---------------------------------
|   |   |   |   | Q |   |   |   |
---------------------------------
|   |   |   |   |   |   |   |   |
---------------------------------
**********************************************
```

Sample output:

```
valid solution
incorrect attempt
valid solution
incorrect attempt
```