HSPC 2019 PROBLEMS

1 QUALITY CONTROL

The toy company only sells blue colored toys. The quality control engineer has set up a machine that spot checks each item in several locations (from 2 to 10 locations per item), and it reports the color at each location. For an item to pass quality control, it must be blue in at least 50% of the locations. The first line of input contains the number of items to check. Each other line contains the colors detected in a single item. The number of spot checks will not be the same for all items. The machine will decide that number as shown in the example below. The colors are separated by a space. You can assume that colors will be spelled with lower case letters (a–z). For each item, output whether the item "passed" or "failed" the quality control check.

Example 1:

Input

5
blue red green
blue blue orange red
red green green red blue blue blue red
blue blue red
red orange blue

Output

failed
passed
failed
passed
failed

2 FRUITS

The goal of this program is to create a new order processing system. The system reads pricing information for the various delicious varieties of fruit stocked by a company, and then processes invoices from customers, determining the total amount for each invoice based on the type and quantity of fruit for each line item in the invoice. The program input starts with the pricing information. Each fruit price (single quantity) is specified on a single line, with the fruit name followed by the price. You can assume that each fruit name is a single word consisting of alphabetic characters, spelled in lower case letters. You can also assume that prices will have exactly two decimal places after the decimal point. Fruit names and prices are separated by a single space character. A single line consisting of the text END_PRICES (as shown in the example below) terminates the list of fruit prices. After the fruit prices, there will be one or more invoices. Each invoice consists of a series of line items. A line item is a fruit name followed by an integer quantity, with the fruit name and quantity separated by a single space. You can assume that no line item will specify a fruit name that is not specified in the fruit prices. A line consisting of the text END_INVOICE terminates each invoice. A QUIT line will always terminate the overall input. You can assume that if an invoice has started it will end with END_INVOICE and QUIT will appear only after the invoice is ended.
For each invoice, the program should print a single line of the form
Total: X.YY, where X.YY is the total cost of the invoice, which is the sum of the costs of all of the line items in the invoice. The cost should be printed with exactly two digits after the decimal point.

Example
Input
orange 0.80
pomegranate 2.50
plum 1.20
peach 1.00
persimmon 1.75
lime 0.60
END_PRICES
persimmon 2
orange 3
peach 1
plum 10
pomegranate 5
END_INVOICE
peach 11
plum 5
orange 1
lime 9
END_INVOICE
QUIT

Output
Total Invoice 1: 31.40
Total Invoice 2: 23.20

3 EXPRESSIONS

Write a program that evaluates infix expressions involving integer values. The evaluator should support the following operators: + (addition), - (subtraction), and * (multiplication), and / (division). It should also allow parentheses to be used for grouping. Division is integer division (meaning that division operations should discard any remainder.) The input will consist of one or more lines, such that each line specifies one infix expression. Each token (integer value, operator, or parenthesis) is separated by a space. The end of the program is specified by the text QUIT. You can assume that every expression will be valid. For each input expression, the program should print a single line containing the integer result of evaluating the expression.

Hints and specifications:

- The + and - operators have equal precedence
- The * and / operators have equal precedence, and have higher precedence than + and –
- All of the operators are left associative: e.g., 25 - 9 - 3 means (25 - 9) - 3

Example

Input

```
4
( 4 )
23 / 3
3 + 4 * 5
( 3 + 4 ) * 5
25 - 9 - 3
QUIT
```

Output (corresponding to the example input shown above):

```
4
4
7
23
35
13
```

4 SPLOTVIAN CUISINE

For a Splotvian twist on any dish, just add chips! In order to help restaurants update their menus to cater to fans of Splotvian cuisine, you decide to write a program that will take the name of any dish and add "and chips" to it. An exception occurs for dishes whose names start with "q". Each line of input is the name of a menu item. As a special case, if a line consists of the text "DONE", the program should exit immediately without any further output. For each input line (other than "DONE"), the program should print the same line, but with "and chips" added (with a space before "and"), unless the name of the dish starts with "q", in which case the name of the dish should be printed verbatim.

Example
Input
pie
biryani
steak tartare
chicken cordon bleu
quinoa with wild mushrooms
jellied eels
baked alaska
cherries jubilee
DONE

Output
pie and chips
biryani and chips
steak tartare and chips
chicken cordon bleu and chips
quinoa with wild mushrooms
jellied eels and chips
baked alaska and chips
cherries jubilee and chips

5 QUADRATIC

Consider the quadratic equation below.

$$ax^2+bx+c=0$$

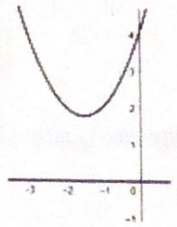
As you know, this can be solved via the quadratic formula.

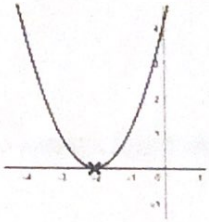
$$x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

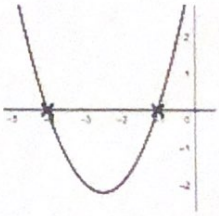
The portion under the square root sign is the discriminant D.

$$D=b^2-4ac$$

The discriminant being negative, zero, or positive dictates whether the equation has zero, one, or two real number solutions, respectively. A real number is any number on the number line, as opposed to a complex/imaginary number like 3 - 4i.







If the discriminant < 0, then there are complex number solutions (Graph 1)
If the discriminant = 0, then there is one real number solution (Graph 2)
If the discriminant > 0, then there is two distinct real number solutions (Graph 3)

Write a program that asks the user to provide a value of a, b, and c. The program should indicate how many solutions the quadratic has, and if they are real, what the solutions are. Note: The judges will only test the program for real number solutions. The output should be formatted for 2 decimal places.

Example
Input
Enter a: 5
Enter b: 10
Enter c: 2

Output
This equation has two distinct real number solutions.
X 1: -0.11
X 2: -1.8

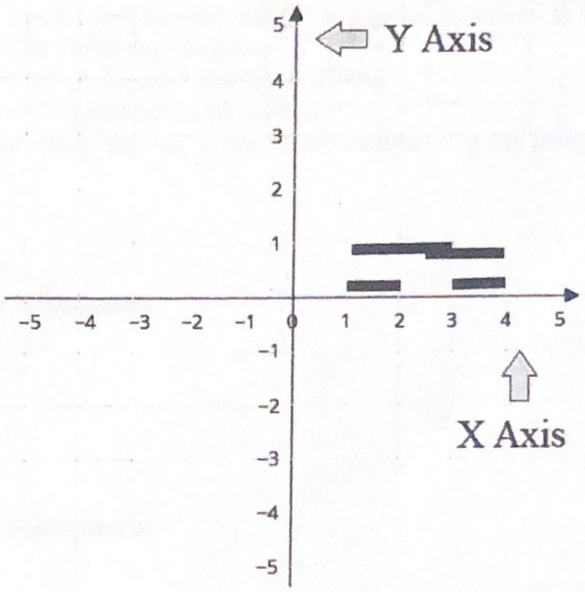
In case of one real number solution, the output should print the solution in this format:
X 1: .01

6 OVERLAP

Ask the user for two number intervals. Write a program to print out whether the two intervals overlap or not.
Interval 1 - [1, 2] – Green line
Interval 2 - [3, 4] – Orange line
There is no overlap between the two intervals

Interval 1 - [2.5, 4] – Black line
Interval 2 - [1, 3] – Blue line
There is an overlap of .5 units between the two intervals

Interval 1 - [1, 3] – blue line
Interval 2 - [3, 4] – Orange line
The points overlap at one point.



Example 1
Input
Enter Interval 1: 2 3
Enter Interval 2: 4 5
Output
There is no overlap between the two intervals

Example 2
Input
Enter Interval 1: 2 6
Enter Interval 2: 4 5
Output
There is an overlap of 1 units between the two intervals

7 GIFT SEASON

It is the gift-giving season. There are 10 students in a classroom assigned numbers 1 through 10. Each student is RANDOMLY assigned someone to give a gift. However, this must be done so that there is one large cycle of gift giving at the end. Example of valid and invalid cycles are shown in the table below.

Scenario 1			Scenario 2			Scenario 2	
Valid			Valid			Invalid	
From	to		From	to		From	to
1	6		4	8		4	8
6	3		8	6		8	6
3	4		6	7		6	4
4	5		7	5		7	5
5	2		5	2		5	2
2	7		2	10		2	10
7	8		10	3		10	3
8	9		3	9		3	9
9	10		9	1		9	1
10	1		1	4		1	7

Example 1

Input

Enter the pairs (from, to):

1 6
6 3
3 4
4 5
5 2
2 7
7 8
8 9
9 10
10 1

Output

This is a valid gift giving cycle

Example 2

Input

Enter the pairs (from, to):

4 8
8 6
6 4
7 5
5 2
2 10
10 3
3 9
9 1
1 7

Output

This is an invalid gift giving cycle

8 ROLLER COASTER

You are one of the world's premier designers of two-dimensional (2D) roller coasters. You can simulate a 2D roller coaster physics, and be sure that your riders reach the end of the track, and reach it alive. The notation for describing a 2-D coaster used here is explained below.

_/__/\

Each \ character indicates a downward slope, each _ character indicates a level grade, and each / character indicates an incline. The car moves from left to right, starting at the leftmost position. Its velocity is initially zero. Each downward slope increases the car's velocity by one. Each incline decreases the car's velocity by one. Level grades do not change the car's velocity. The car **only** moves forward if its velocity is positive. The simulation tracks the car's velocity and location. If the car moves past the last position with a positive or zero velocity, then it is considered to reach the end of the track. However, if the car's final velocity is 10 or greater, the car is considered to have crashed, meaning the riders did not survive.
The input contains a 2-D roller coaster specification using the notation described above. The program should print a single line of output with one of the following responses:
• Whee! -- If the car reached the end of the track and did not crash
• Splat! -- If the car reached the end of the track and crashed
• Stopped at N if the car did not reach the end of the track -- where N is the position where the car stopped (the leftmost position being 1)

Example 1
Input
Enter the specifications for 2-D rollercoaster:
_/__/\
Output
Whee!

Example 2
Input
Enter the specifications for 2-D rollercoaster:
__//_//_
Output
Stopped at 11

Example 3
Input
Enter the specifications for 2-D rollercoaster:
_/_//___
Output
Splat!

9 FOOTBALL SCORING

We wish to determine the number of ways of achieving a given score in football. For the purposes of this program, we assume that there are five ways of scoring:

- A safety: 2 points
- A field goal: 3 points
- A touchdown with no extra points: 6 points
- A touchdown with one extra point: 7 points
- A touchdown with two extra points: 8 points

For example, there are four ways to score 9 points:

1. 3 safeties and 1 field goal
2. 1 safety and 1 touchdown with one extra point
3. 1 field goal and 1 touchdown with no extra point
4. 3 field goals

Your program should take as input a single nonnegative integer score no greater than 100, and it should output the number of ways of achieving that score.

Example 1
Input
Enter score: 9
Output
Number of ways: 4

Example 2
Input
Enter score: 0
Output
Number of ways: 1

Example 3
Input
Enter score: 6
Output
Number of ways: 3

10 TIC TAC TOE

The numbers in a magic square can represent the spaces on a tic-tac-toe board.

6	7	2
1	5	9
8	3	4

The advantage is that the sum of any row, column, or diagonal is the same. Therefore, any winning combination has values that sum to same number, in this case 15. Given three "X" moves and three "O" moves, determine if "X" or "O" has already won. If not, does "X" have a winning move or if not, does "X" have a blocking move. Blocking move means that we can stop the other player from winning.

Example 1

Input

Enter first X move: 6
Enter first O move: 9
Enter second X move: 1
Enter second O move: 7
Enter third X move: 5
Enter third O move: 8

Output

Play 4 to win

Example 2

Input

Enter first X move: 6
Enter first O move: 1
Enter second X move: 5
Enter second O move: 8
Enter third X move: 4
Enter third O move: 3

Output

X has already won