(Heron's formula) Write a C program to find the area of a triangle using Heron's Formula, which calculate the area of a triangle using the three side lengths of the triangle. If we know the length of all sides of any triangle, then we can calculate the area of triangle using Heron's Formula. Heron's formula is a generic formula and is not specific to any triangle, it can be used to find area of any triangle whether it is right triangle, equilateral triangle or scalene triangle. Heron's formula relates the side lengths, and the perimeter to the area of a triangle.

Let A be the area of a triangle, a, b and c be the length of the three sides of the triangle, and s be the semi-perimeter of the triangle. Note that if the given side lengths can't form a triangle, print "It is not a triangle!".

$$s = \frac{a+b+c}{2}$$

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Input:

- Input three number representing the length of the three sides of a triangle.
- Input negative number to end the program.

Output:

- Print the area of the triangle.
- The answer should round to 5 decimal places.

Sample input:

3 4 5 0 0 180 -1

Sample output:

6.00000
It is not a triangle!

Taipei City Government would like to develop a system for taxi fare estimation. The system can calculate the fare that the passengers have to pay for a given mileage data, or the mileage they can run for a given taxi fare.

The system will read and parse the data from a file, namely predict.txt, and calculate the price or mileage according to the following rules. Finally, output the result on the screen.

If the result is taxi fare, please round to a whole number (an integer).

Rule:

Daytime fare : $70 for 1.25km, $5 for every 0.2km after 1.25km

Nighttime fare : $90 for 1.25km, $5 for every 0.2km after 1.25km

Note:

1. If the extra distance is less than 0.2km, you do not need to pay the money (e.g. 1.26km=$70).

The input data format is as follows: (A/B) (Y/N) (Fare/mileage).

A/B: "A" is the option for the case that your program calculates the taxi fare for a given mileage.
"B" is the option for the case that your program calculates the mileage for a given input fare.

Y/N: "N" means to calculate the Daytime fare, "Y" means to calculate the Nighttime fare.

Price/Mileage: The input fare or mileage data.

predict.txt format(input):	output:
B Y 10	You don't have enough money.
A Y 1.25	$90
A N 5.2	$165
B Y 100	1.65km
B N 69	You don't have enough money.

(Complex number arithmetic) Write a program that calculates the addition, subtraction, and multiplication of two complex numbers. The first input value specifies the calculation mode, where 1 represents addition, 2 represents subtraction, 3 represents multiplication, and -1 represents the end of program. The program then takes two complex numbers as inputs and prints the corresponding result. The program shouldn't stop until the user inputs -1.

Hint:

$i = \sqrt{-1}$

$i^2 = -1$

$(\mathrm{a} + \mathrm{b}i) + (\mathrm{c} + \mathrm{d}i) = (\mathrm{a} + \mathrm{c}) + (\mathrm{b} + \mathrm{d})i$

$(\mathrm{a} + \mathrm{b}i) - (\mathrm{c} + \mathrm{d}i) = (\mathrm{a} - \mathrm{c}) + (\mathrm{b} - \mathrm{d})i$

$(\mathrm{a} + \mathrm{b}i) \times (\mathrm{c} + \mathrm{d}i) = (\mathrm{a} \times \mathrm{c} - \mathrm{b} \times \mathrm{d}) + (\mathrm{a} \times \mathrm{d} + \mathrm{b} \times \mathrm{c})i$

Sample Input:

1
2.5+6.8i
1.7+1.6i
2
2.7+1.5i
1.5+8.6i
3
1.2+3.4i
2.6+1.7i
-1

Sample Output:

4.20+8.40i
1.20-7.10i
-2.66+10.88i

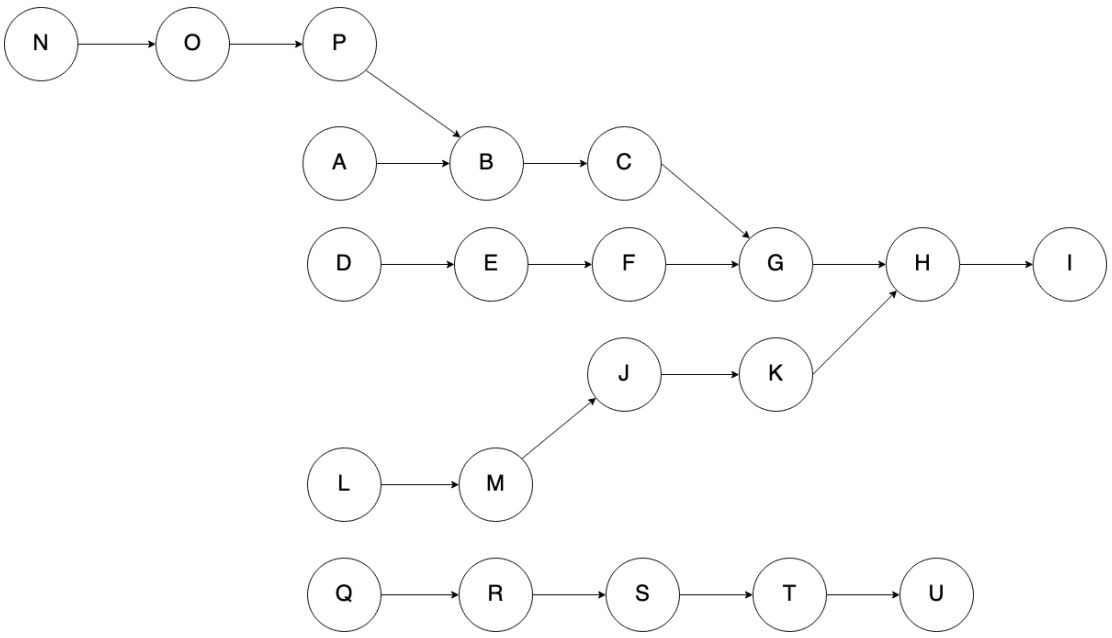
A list is an ordered sequence of characters. In this problem, you will be given a main list and five sub-lists. You need to connect the sub-lists to other lists (either main list or sub-lists). Specifically, if the last character of a sub-list is the same as a character in another list, you can attach the last character in the sub-list to the corresponding character in the other list. For example, in the following case, the main list is 'D E F G H I'. A sub-list 'A B C G' can connect to the main list since the last character 'G' can be attached to the character 'G' in the main list. In this problem, you have to check whether the sub-lists can connect to the main list either directly or through other sub-lists.

In the input, each line is a list. The first line is the main list and the other five lines are the sub-lists. Please write a program to check whether these sub-lists can connect to the main list directly or through other sub-lists.
If a sub-list can connect to the main list, output "True"; otherwise, output "False".

Sample input:	Sample out:
D E F G H I	True
A B C G	True
J K H	True
L M J	True
N O P B	False
Q R S T U	

Explanation:
D E F G H I is the main list.
A B C G is True, because it can connect to the main list.
J K H is True, because it can connect to the main list.
L M J is True, because it can connect to the main list through J K H.
N O P B is True, because it can connect to the main list through A B C G.
Q R S T U is False, because it cannot connect to the main list.

The resulting connection of this example is shown in the figure:



An adventurer discovered a cave with many jewels, but the path inside was very complicated and some paths could not be passed. To avoid getting lost and collect all the gems in the cave, please help him by recording all the paths he walked on and counting the total number of jewels found.

The input file, called "input.txt," contains the map of the cave. The cave is 31x31 in size. In the map, "#" represents a wall, "." represents a path, "O" represents a jewel, "@" represents a path the adventurer has walked on, and "X" represents a path that cannot be passed. The entrance is on the leftmost side of the map. Please write a program that finds all the jewels in the cave and prints out the total number of jewels and the path the adventurer took in the same format of the input file.

The input sample may like :

```
###############################
......#......O......#.........#
#.###.#.##########.#####.###.#
#...#...#...#.....#OO...#.#.#.#
#.#.###.#.#.#.#.#####.#O#O#.#
#.#.#...#.#.#.#.#.....#...#.#.#
#X#.#.###O#.###.#####.#####.#.#
#.#.#O.O#.#...#.....#O#...#O..#
#.#.#####.###X#.###.###.#.#.###
#.#O..#...#.#.#O..#.O...#.#.#.#
#####.#.###.#O#.#####.###.#.#.#
#.O..O#...O.#.#.#O..#...#.#.#.#
#.###.###.###.#.#.#.###.#.#.#.#
#.#.X...#.#...#.#.#.O.#.#.#O#.#
#.#####O#O#.###O#O###O#.#.#.#.#
#.....#O#.#.#.O.#.#.#.#X#.O.#.#
#.###.###.#O#####.#.#.#####.#.#
#.X.#...O.#...#...#.#..O..#.#.#
###.########.#O###.#####.#.#.#
#.#.#.......#.#.#....O..#.#.#.#
#.#.#.####X#.#.#####.#.#.#.#.#
#.#.#O.O..#.#O#.#...#.#O#.#O#.#
#O#.#####.###.#.#.#.#.#.#.#.#.#
#.#.......#...#.#.#..O#.#.#...#
#.#####.###.###.#O#####.#.###.#
#.O...#.#...#...#...#O..#..X#.#
#.#O###O#.###O####O######.#.#.#
#.#.X...#.....#O..#.X.#.....#.#
#.############.#.###.#O######.#
#.X............#.....#......O#
###############################
```

Screen output should like:

```
Total jewel : 25
###############################
@@@@@@#@@@@@@@@@@@@@@#@@@@@@@@@@#
#@####@#@###########@#####@###@#
#@@@@#@@@@#@@@@#@@@@@@#@@@@@@#@#@#@#
#@#@###@#@#@#@#@#@#####@#@#@#@#
#@#@#@@@@#@#@#@#@@@@@@#@@@@#@#@#
#X#@#@###@#@###@#####@#####@#@#
#.#@#@@@@#@#@@@@#@@@@@@#@#@@@@#@@@@#
#.#@#####@###X#@###@###@#@#@###
#.#@@@@#@@@@#@#.#@@@@#@@@@@@#@#@#@#
#####@#@###@#O#@#####@###@#@#@#
#@@@@@@#@@@@@@#.#@#O..#@@@@#@#@#@#
#@###@###@###.#@#.#.###@#@#@#@#
#@#.X@@@@#@#...#@#.#.O.#@#@#@#@#
#@#####@#@#.###@#O###O#@#@#@#@#
#@@@@@@#@#@#.#@@@#.#.#.#X#@@@@#@#
#@###@###@#O####.#.#.#####@#@#
#@X.#@@@@@@#...#...#.#..O..#@#@#
###.########.#O###.#####.#@#@#
#.#.#.......#.#.#....O..#.#@#@#
#.#.#.#####X#.#.#####.#.#.#@#@#
#.#.#O.O..#.#O#.#...#.#O#.#@#@#
#O#.#####.###.#.#.#.#.#.#.#@#@#
#.#.......#...#.#.#..O#.#.#@@@#
#.#####.###.###.#O#####.#.###@#
#.O...#.#...#...#...#O..#..X#@#
#.#O###O#.###O####O######.#@#
#.#.X...#.....#O..#.X.#.....#@#
#.############.#.###.#O######
#.X............#.....#......O#
###############################
```

Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules:

- Each row must contain the digits 1-9 without repetition.
- Each column must contain the digits 1-9 without repetition.
- Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.

Note:

- A Sudoku board (partially filled) could be valid but is not necessarily solvable.
- Only the filled cells need to be validated according to the mentioned rules.
- Input file name is "input.txt".
- If the verification result is feasible, please output "Valid", otherwise output "Invalid".
- Each Sudoku grid is 9x9 in size, and the blank spaces are represented by '.' in the input file.
- The input file can contain a maximum of 10 Sudoku grids, and a minimum of one.

Example:

input.txt

```
5 3 . . 7 . . . .
6 . . 1 9 5 . . .
. 9 8 . . . . 6 .
8 . . . . 6 . . 3
4 . . 8 . 3 . . 1
7 . . . 2 . . . 6
. 6 . . . . 2 8 .
. . . 4 1 9 . . 5
. . . . 8 . . 7 9

8 3 . . 7 . . . .
6 . . 1 9 5 . . .
. 9 8 . . . . 6 .
8 . . . . 6 . . 3
4 . . 8 . 3 . . 1
7 . . . 2 . . . 6
. 6 . . . . 2 8 .
. . . 4 1 9 . . 5
. . . . 8 . . 7 9
```

Screen output:

Sudoku 1: Vaild
Sudoku 2: Invalid