PLU February 2015 Programming Contest

Novice Problems

I. General Notes

1. Do the problems in any order you like.

2. Problems will have either no input or will read input from standard input (stdin, cin, System.in -- the keyboard). All output should be to standard output (the monitor).

3. All input is as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros.

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

II. Names of Problems

Number	Name
Problem 1	Hungry Cat
Problem 2	Climb
Problem 3	Fate
Problem 4	Favorite
Problem 5	Mixed Fractions
Problem 6	MMM
Problem 7	Majority
Problem 8	Play
Problem 9	Radians
Problem 10	Polyhedra
Problem 11	Happy Numbers
Problem 12	Luminance

1. Hungry Cat

Problem Description: Problem Description: Print the ascii art as shown below. The last line contains 20 “dash” characters (i.e., not underscore characters). The eye of the fish is the lower case letter o (as in look). The nose is a dash.

Input: none

Output: Print the picture as shown below.

Sample Input: none

Sample Output:

```
    /\~~/\
   (  ^-^ )
 ~  )    (    (((
(( /      \    )))
 (  ) || ||  >+++o>
--------------------
```

2. Climb

Problem Description: An ant falls into an ant lion trap with a slippery sand slope and is bound and determined to climb out to safety to avoid certain death in the jaws of the predator below. Fortunately for the ant, the quality of the sand for this trap is not quite up to "certain death" standards, allowing it to escape, eventually.

However, for every X millimeters of progress upwards, the sand causes the ant to slip back down Y millimeters. Given the depth of the hole in M millimeters, determine how many climbing attempts it will take for the ant to reach the top edge of the hole to safety. Also calculate the total distance, upwards and downwards, the ant has traversed in its attempt to reach the top.

For example, if the ant is at the bottom of an 11 mm hole, and climbs up 5 millimeters each time, only to slip back 3 mm because of the loose sand, the progress would be +5, -3, +5, -3, +5, -3, and finally +5 to reach the top. This effort took 4 climbing attempts and covered a total distance of 29 mm.

Assumptions:
1. The ant will reach the top, eventually.
2. X will always be greater than Y
3. M will always be greater than X
4. The ant has unlimited energy, no matter how deep the hole or how fierce the ant lion.

Input: An initial integer value N, followed by N sets of data, each on one line. Each set of data consists of three integers X, Y, and M, as described above.

Output: Two integers C and D, C being the number of climbing attempts required, and D being the total distance traversed to reach the top.

Sample Input:
```
3
5 3 11
7 2 27
2 1 13
```

Sample output:
```
4 29
5 43
12 35
```

3. Fate

Problem Description: Sherry is playing a new game called Fate. This is a card game where the goal is to get as many duplicates in your hand as possible. There will only be five different cards in this game labeled A, B, C, D, E.

The game starts by dealing each player five cards. For each turn, the player is dealt a card and has to choose a card to remove from the hand such that there will always be only five cards in the hand at a time.

Sherry has decided to apply a simple algorithm to her card-keeping choices. After a new card is added to the hand, she removes the card that has the least number of occurrences. If there are two different valued cards that have the same number of occurrences then the one of lower value is eliminated. The value order for the cards is A < B < C < D < E.

For example, the input string **`A A C D E A D D`** means the hand starts with 5 cards **`A A C D E`** and that the cards **`A D D`** will follow in that order. After receiving the 6th card, **`A`**, her hand will have an equal number of occurrences of C, D, and E. Thus, C is removed since it is the least in value. The next card received is **`D`**, and since **`E`** has the least number of occurrences it is removed. When **`D`** is received, **`A`** is removed since **`A`** and **`D`** have the same number of occurrences, The final hand for this example is **`A A D D D`**.

Input: A string of characters representing the cards that Sherry has received. The first 5 characters are the cards that Sherry is dealt, and all the following cards are received applying her algorithm.

Assumptions: The input string will have at least 5 characters.

Output: Output the final hand for each string of characters in order from least to greatest value.

Sample Input
```
A B C D E A D D
E E B E E C C C C C D D
A A A A A A B C D E
```

Sample Output
```
A A D D D
D E E E E
A A A A A
```

4. Favorite

Problem Description: Introduce your team to the judges by outputting your first names and your favorite movie, TV, or stage actor or actress in a sentence, as shown below. The sentence structure must match EXACTLY, word for word, punctuation, spelling, gender pronoun, etc., except for your name and the name of your favorite actor or actress.

Input: none

Sample Output:
```
My name is John, and my favorite movie actor is John Wayne.
My name is Sarah, and my favorite movie actress is Meryl Streep.
My name is Mark, and my favorite movie actor is Al Pacino.
```

5. Mixed Fractions

Problem Description: You are part of a team developing software to help students learn basic mathematics. You are to write one part of that software, which is to display possibly improper fractions as mixed fractions. A proper fraction is one where the numerator is less than the denominator; a mixed fraction is a whole number followed by a proper fraction. For example the improper fraction 27/12 is equivalent to the mixed fraction 2 3/12. You should not reduce the fraction (i.e. don't change 3/12 to 1/4).

Input: Input has one test case per line. Each test case contains two integers in the range [1, $2^{31}-1$]. The first number is the numerator and the second is the denominator. A line containing 0 0 will follow the last test case.

Output: For each test case, display the resulting mixed fraction as a whole number followed by a proper fraction, using a single space to separate output tokens.

Sample Input:
```
27 12
2460000 98400
3 4000
0 0
```

Sample Output:
```
2 3 / 12
25 0 / 98400
0 3 / 4000
```

6. MMM

Problem Description: Measures of central tendency traditionally are so much a part of data analysis that they are a common part of any student's mathematical training in school. The most common measures are **mean, median,** and **mode**, which are simple enough to calculate. Each is defined below.

The "**mean**" is the "average" you're used to, where you add up all the numbers and then divide by the number of numbers.

The "**median**" is the "middle" value in the list of ordered numbers. To find the median, your numbers have to be listed in numerical order. If there are an even number of numbers, the median is the mean (i.e., average) of the two middle numbers.

The "**mode**" is the value that occurs most often. If no number is repeated, then there is no mode for the list.

For example, the list 1 3 5 5 has mean 7, median 4, and mode 5.

We're going to take it a step further and come up with the **MMM** measure, which is the average of the three, i.e., **(mean+median+mode)/3.** The MMM measure may or may not have any validity for serious data analysis, but it makes for an interesting programming problem, for sure.

Assumptions: It is guaranteed for each data set that there will be a clear and unique mode.

Input: Several lines of positive real numbers, each separated by a single space.

Output: For each line of values, find the mean, median, mode, and MMM measure as described above, and output each value to two decimal places, each separated by a single space.

Sample input:
```
4 2 3 7 8 6 4 9.1 4.5 8
4 7 6 2.3 56 7 12 23.6 7 16 4
1 2.3 4 5.6 7 8.9 2.3 9.8 6.5 2 3
```

Sample output:
```
5.56 5.25 4.00 4.94
13.17 7.00 7.00 9.06
4.76 4.00 2.30 3.69
```

7. Majority

Problem Description: The votes are in! Mathematicians world-wide have been polled, and each has chosen their favorite number between 1 and 1000. Your goal is to tally the votes and determine what the most popular number is. If there is a tie for the greatest number of votes, choose the smallest number with that many votes.

Input: Input will start with a single line containing the number of test cases, between 1 and 100 inclusive. For each test case, there will be a single line giving the number of votes V, $1 \leq V \leq 1000$. Following that line will be V lines, each with a single integer vote between 1 and 1000.

Output: The most popular number.

Sample input:
```
3
3
42
42
19
4
7
99
99
7
5
11
12
13
14
15
```

Sample output:
```
42
7
11
```

8. Play

Problem Description: The play button icon on most computer video or audio programs these days is shaped like an isosceles triangle, such as this one.

Your job is simple. Given an integer indicating the size of the play button, output a star pattern in the shape of the play button, as shown in the sample output.

Input: Several integers x ($2 \leq x \leq 10$) all on one line, each separated by a single space.

Output: The Play button shape using a 2D pattern of stars, with a blank line separating each output.

Sample input:
```
3 4 5
```

Sample output:
```
*
***
*****
***
*

*
***
*****
*******
*****
***
*

*
***
*****
*******
*********
*******
*****
***
*
```

9. Radians

Problem Description: Input an angle measure in degrees and output the equivalent in radians in terms of PI. If the radian measure is 1PI, output only PI. If the radian measure in terms of PI is a whole number other than 1, output the number with no decimal places, followed by PI, otherwise output the decimal value expressed to two places, followed by PI.

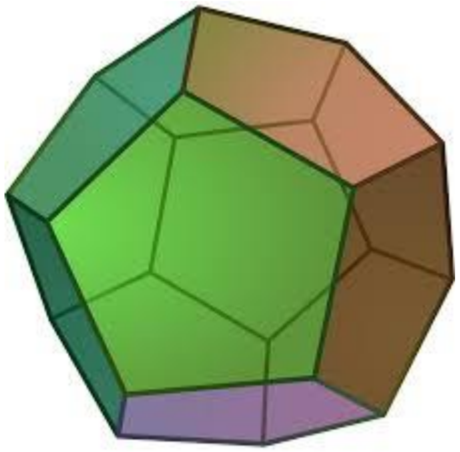
Input: Several integers, each on one line, representing angle measures in degrees.

Output: The radian measure in terms of PI, as described above.

Sample Input:
```
90
180
360
45
```

Sample output:
```
90 degrees = 0.50PI radians
180 degrees = PI radians
360 degrees = 2PI radians
45 degrees = 0.25PI radians
```

10. Polyhedra



Problem Description: Given a sphere, you can slice through the surface of the sphere to make different convex polyhedra. All of these convex polyhedra have the Euler characteristic which can be defined as follows:

$$V - E + F = 2$$

where V represents the number of vertices, E the number of edges and F the number of faces on a convex polyhedron.

Input: Input begins with a line with a single integer T, $1 \leq T \leq 100$, denoting the number of test cases. Each test case consists of a single line with two space-separated integers V and E ($4 \leq V, E \leq 100$), representing the number of vertices and edges respectively of the convex polyhedron.

Output: For each test case, print on a single line the number of faces in the defined polyhedron.

Sample Input:
```
2
8 12
4 6
```

Sample output:
```
6
4
```

11. Happy Numbers

Problem Description: A happy number is a number that has the following property: start with the number and replace it with the sum of the squares of its digits. Repeat the process with the result. If the number eventually becomes 1, the starting number is a happy number (along with all numbers in the sequence). If the process does not reach a value of 1, the number is a sad number.

For example, 320 is a happy number, as shown by the following sequence:

$320 \rightarrow 3^2 + 2^2 + 0^2 = 13$
$13 \rightarrow 1^2 + 3^2 = 10$
$10 \rightarrow 1^2 + 0^2 = 1$

If a number is sad number, its sequence will eventually reach a repeating cycle of numbers that includes 4.

Input: The first line contains a single integer N representing the number of values to test. It is followed by N lines, each containing a positive integer to test for happiness.

Output: For each value, display the number followed by a happy emoticon if the number is a happy number, otherwise, show a sad emoticon. There should be a single space between the number and the emoticon.

Sample Input:
```
5
320
908
13
10
33
```

Sample Output:
```
320 :-)
908 :-(
13 :-)
10 :-)
33 :-(
```

12. Luminance

Problem Description: Relative luminance is a way of measuring the perceived brightness of a color on a computer monitor. A common formula for computing relative luminance Y of an (R, G, B) color is the following:

$Y = 0.2126 R + 0.7152 G + 0.0722 B$

The variables R, G, and B are the red, green and blue intensities (represented by values between 0.0 and 1.0). For each set of R, G, and B values you will need to determine if the data set is valid or invalid. A data set (R, G, B) is invalid if any one of them is outside the range 0 to 1 inclusive. You will also be given a threshold, T, and if the data set is valid you will need to determine if the relative luminance is greater than or equal to the threshold. Your program will calculate the following:

- The number of valid data sets
- The average luminance of the valid data sets
- The number of valid data sets that have a relative luminance greater than or equal to the threshold.
- The number of invalid data sets

Input: The first line is an integer, N, representing the number of (R, G, B) data sets. The second line is the threshold value, T, a floating point number. The threshold is followed by N lines. Each line contains one (R, G, B) data set, represented by three floating point numbers separated by spaces. Note that the values are not necessarily guaranteed to be between 0.0 and 1.0. If any color has an R, G, or B value that is outside the range 0 to 1 inclusive, it should be ignored except for the invalid count.

Output: Display the number of valid colors, the average relative luminance of the valid color values (displayed with 3 digits to the right of the decimal), the number of color values that have a relative luminance that is greater than or equal to the threshold value (T) (excluding invalid colors), and the number of invalid colors.

Sample Input:
```
5
0.5
0.5 0.5 0.1
0.2 0.9 0.0
0.85 0.2 0.521
0.1 0.2 0.3
0.0 10.2 0.0
```

Sample Output:
```
4
0.426
1
1
```