PLU February 2014 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. 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.

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

Number	**Name**
Problem 1	NFC West vs North
Problem 2	Cats
Problem 3	Dogs
Problem 4	Finding Nemo
Problem 5	Palindromes
Problem 6	Dominant Species
Problem 7	Opposite Words
Problem 8	Magic Square
Problem 9	Sharing
Problem 10	Coupons
Problem 11	Ship Selection
Problem 12	Federation Favorites

Good luck!

1. NFC West vs North

Problem Description: Your friends from the Midwest are always reminding you that the Seattle Seahawks won the NFC West Division with a losing record of 7-9 in 2011. At the time they were the only team to go to the NFL playoffs with a losing record (in a full season). To put an end to this nonsense you have decided to write a program to print the current NFC West and NFC North division standings.

Input: none

Output: Print the NFC West and NFC North division records as shown below. The character "−", is a dash not an underscore.

Sample Input: none

Sample Output:

```
NFC West       W   L  T
----------------------
Seattle        13  3  0
San Francisco  12  4  0
Arizona        10  6  0
St. Louis      7   9  0

NFC North      W   L  T
----------------------
Green Bay      8   7  1
Chicago        8   8  0
Detroit        7   9  0
Minnesota      5  10  1
```

2. Cats

Problem Description: Print the ascii cat as shown below.

Input: none

Output: Print the picture as shown below. The horizontal line __ is comprised of two underscore characters.

Sample Input: none

Sample Output:

3. Dogs

Problem Description: Print the ascii dog as shown below.

Input: none

Output: Print the picture as shown below. The long horizontal line __ is comprised of two underscore characters, and the dog's nose is a zero.

Sample Input: none

Sample Output:

```
|\_/|
|q p|   /}
( 0 )"""\
|"^"`    |
||_/=\\__|
```

4. Finding Nemo

Problem Description: You must determine if the word "Nemo" (not case sensitive) is hidden in each line of input. The letters must be consecutive, but capitalization is not important.

Input: The input will contain multiple problem sets. Each problem set consists of a single line containing no more than 80 characters. The end of the input is denoted by the string "EOI". The string "EOI" will be on a single line by itself and is not a problem set.

Output: For each problem set print "Found" if the word "Nemo" is hidden in the line, or print "Missing" if the word "Nemo" is not in the line.

Sample Input:

```
Marlin names this last egg Nemo, a name that Coral liked.
While attempting to save nemo, Marlin meets Dory,
a good-hearted and optimistic regal blue tang with short-term memory loss.
Upon leaving the East Australian Current,(888*%$^&%0928375)Marlin and Dory
NEMO leaves for school and Marlin watches NeMo swim away.
EOI
```

Sample Output:

```
Found
Found
Missing
Missing
Found
```

5. Palindromes

Problem Description: A palindrome is a word, phrase, number, or sequence of characters that is same in either the forward or reverse direction. Typically one does not worry about capitalization, but all characters including blanks are considered when determining if a string is a palindrome. The following are all palindromes:

```
Anna
Harrah
Arora
Nat tan
9998999
123  321
$$$&&$$$
```

Write a program to determine which lines are palindromes.

Input: The first line of input will be a positive integer, **n**, indicating the number of problem sets. Each problem set is a single line of text, and there will be no blank lines.

Output: For each problem set (i.e., line of input) print "Yes" if the string is a palindrome and "No" if it is not a palindrome.

Sample Input:

```
6
Nat tan
Palindrome
123454321
Dogs and Cats
**()()**
1 221
```

Sample Output:

```
Yes
No
Yes
No
No
No
```

6. Dominant Species

Problem Description: The Natural Wildlife Federation is trying to determine the dominant species in several different areas in the United States. They have sent researchers out to record the current population of four different predators (Bobcat, Coyote, Mountain Lion, and Wolf). They will use this information to determine which species dominates a particular ecological location. Because the animals differ in size, weight, and hunting ability each species has been given a weighted value as follows:

Species	Value
Bobcat	2
Coyote	1
Mountain Lion	4
Wolf	3

Thus, one Mountain Lion is equivalent to two Bobcats and one Wolf is equivalent to three Coyotes. The researchers have recorded animal sightings by writing a B for bobcat, C for coyote, M for mountain lion, and W for wolf. For example, if the researcher at Saguaro saw 5 wolves, 10 coyotes, 2 bobcats, and no mountain lions they might record this with the string "Saguaro WCCCBCCCCWWBCCWWC"

They need your help to process the raw data and determine the dominant species.

Input: The first line of input will be a positive integer, **n**, indicating the number of problem sets to follow. Each problem set is comprised of one line of text. Each line will contain two "words". The first word will be the location and the second word will be the species count as described above. The second word will have no blanks and only contain the letters 'B', 'C', 'M', and 'W'.

Output: For each problem set print one of the two followings statements:

```
<x>: The <y> is the dominant species
<x>: There is no dominant species
```

In the lines above the place holder <x> should be replaced with the location, and the place holder <y> should be replace with either Bobcat, Coyote, Mountain Lion, or Wolf, where the species selected is the dominant species.

Sample Input:

```
3
Saguaro WCCCBCCCCWWBCCWWC
Sequoia CMCCWMWBMCC
Yellowstone MCCBWB
```

Sample Output:

```
Saguaro: The Wolf is the dominant species
Sequoia: The Mountain Lion is the dominant species
Yellowstone: There is no dominant species
```

7. Opposite Words

Problem Description: One of the listeners of the NPR Sunday Puzzle show described the following puzzle: The word "wizard" has the peculiar property that its letters can be grouped in pairs,

A and Z
B and Y
C and X
…
M and N

that are opposite each other in the alphabet. That is, A and Z are at opposite ends of the alphabet, C and X are three letters in from their respective ends, and M and N are 13 letters in from their respective ends. Can you name a well-known brand name in six letters that has this same property?
Answer: La-Z-Boy (Notice the alphabetic opposites don't have to appear in opposite positions in the word)

Given a list of words you must determine which of the words are "opposite" words. Note that capitalization is not important when determining is a word is an "opposite" word.

Input: The first line of input will be a positive integer, **n**, indicating the number of problem sets that follow. Following the positive integer will be **n** text strings, one per line.

Output: For each problem set print "Yes" if the string is an "opposite" word and "No" if it is not an "opposite" word.

Sample Input:

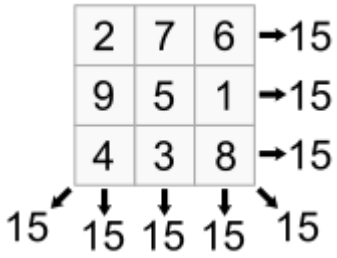
```
5
Wizard
Mark
La-Z-Boy
Love
Hello
```

Sample Output:

```
Yes
No
Yes
Yes
No
```

8. Magic Squares

Problem Description: A magic square is an arrangement of integers in a square grid, where the numbers in each row, and in each column, and the numbers on each main diagonal, all add up to the same value. A magic square has the same number of rows and columns and we will let **m** represent the size (number of rows and columns) of the magic square. Thus, a magic square of size m will have a total of m^2 integers. The following is a magic square with m=3.



You will write a program that will determine if an mxm square is a magic square.

Input: The first line of input will be a positive integer, **n**, indicating the number of problem sets (i.e., magic squares) to follow. Each problem set starts with the integer, m, that specifies the size of the magic square. The next m lines each contain m integers and the integers are separated by one or more spaces.

Output: For each problem set if the square is a magic square print "Magic square of size <m>", where <m> is replaced with the size (number of rows) of the magic square. If the square is not a magic square print "Not a magic square"

Sample Input:

```
4
3
2 7 6
9 5 1
4 3 8
2
14 22
26 10
4
16 2  3  13
5  11 10 8
9  7  6  12
4  14 15 1
2
5 5
5 5
```

Sample Output:

```
Magic square of size 3
Not a magic square
Magic square of size 4
Magic square of size 2
```

9. Sharing

Problem Description: It's after Halloween and you have to share with your sibling(s) because your mother said so! You need to share evenly with your siblings, and your dad gets any leftovers. Given the number of candy pieces and the number of siblings, find how many pieces of candy you get as your share and how many pieces your dad gets.

Input: The first line consists of the number of data sets. Each subsequent line contains two numbers: the number of pieces of candy and the number of siblings (at least 2).

Output: Print
`"You get __ piece(s) and your dad gets __ piece(s)."`

Sample Input:
```
5
22 3
15 5
99 8
7 4
101 5
```

Sample Output:
```
You get 7 piece(s) and your dad gets 1 piece(s).
You get 3 piece(s) and your dad gets 0 piece(s).
You get 12 piece(s) and your dad gets 3 piece(s).
You get 1 piece(s) and your dad gets 3 piece(s).
You get 20 piece(s) and your dad gets 1 piece(s).
```

10. Coupons

Problem Description: You have some coupons that are 20% off any item. Given a price of an item, calculate the new price.

Input: The first line consists of the number of data sets. Each subsequent line contains one price.

Output: Show each price with a dollar sign to the nearest cent.

Sample Input:

```
3
100.00
59.99
20.00
```

Sample Output:

```
$80.00
$47.99
$16.00
```

11. Ship Selection

Problem Description: When Starfleet headquarters gets a request for an exploration expedition, they need to determine which ship from those currently docked in the docking bay to send. They decide to send whichever ship is currently able to make the expedition based on how much fuel is currently stored on the ship as well as how long it will take the ship to arrive at the expected destination. Due to the age and current maintenance of the ships, each ship travels at a different top speed and has a different fuel consumption rate. Each ship reaches its top speed instantaneously.

Input: Input begins with a line with one integer T ($1 \leq T \leq 50$) denoting the number of test cases. Each test case begins with a line with two space-separated integers N and D, where N ($1 \leq N \leq 100$) denotes the number of ships in the docking bay and D ($1 \leq D \leq 10^6$) denotes the distance in light-years to the expedition site. Next follow N lines with three space-separated integers v_i, f_i, and c_i, where v_i ($1 \leq v_i \leq 1000$) denotes the top speed of ship i in light-years per hour, f_i ($1 \leq f_i \leq 1000$) denotes the fuel on ship i in kilos of deuterium, and c_i ($1 \leq c_i \leq 1000$) denotes the fuel consumption of ship i in kilos of deuterium per hour.

Output: For each test case, print a single integer on its own line denoting the number of ships capable of reaching the expedition site. Be careful with integer division!

Sample Input:

```
2
3 100
52 75 10
88 13 44
56 9 5
2 920368
950 950 1
943 976 1
```

Sample Output:

```
2
1
```

12. Federation Favorites

Problem Description: En route to Rigel 7, Chief Engineer Geordi Laforge and Data were discussing favorite numbers. Geordi exclaimed he preferred Narcissistic Numbers: those numbers whose value is the same as the sum of the digits of that number, where each digit is raised to the power of the number of digits in the number.

Data agreed that Narcissistic Numbers were interesting, but not as good as his favorite: Perfect Numbers. Geordi had never heard of a Perfect Number, so Data elaborated, "A positive integer is said to be Perfect if it is equal to the sum of its positive divisors less than itself. For example, 6 is Perfect because 6 = 1 + 2 + 3."

Geordi began thinking about an algorithm to determine if a number was Perfect, but did not have the raw computing ability of Data. He needs a program to determine if a given number is Perfect.

Help Geordi write that program.

Input: Input consists of a single entry per line. Each line contains a single positive integer n, where $2 < n < 100000$ for each case. A line containing -1 denotes the end of input and should not be processed.

Output: For each case, determine whether or not the number is Perfect. If the number is Perfect, display the sum of its positive divisors less than itself. The ordering of the terms of the sum must be in ascending order. If a number is not Perfect, print "<NUM> is NOT perfect." where <NUM> is the number in question. There must be a single space between any words, symbols, or numbers in all output, with the exception of the period at the end of the sentence when a number is not perfect.

Sample Input:

```
6
12
28
-1
```

Sample Output:

```
6 = 1 + 2 + 3
12 is NOT perfect.
28 = 1 + 2 + 4 + 7 + 14
```