RMC Western Montana High School Programming Contest

Morning Session, March 21, 2015

Elvish Rune

The spiked Y elvish rune looks like an ordinary Y except that the bottom stem of the letter carries through until it's even with the top of the rest of the strokes. Create a program that reads in a single positive odd integer and uses character art to create a spiked Y rune that has the same width and height as n. For "background" you can either draw spaces or use the period character. For the "foreground" use asterisks.

Sample input 0:

```
7
```

Sample output 0:

```
*..*..*
.*.*.*
..***
...*
...*
...*
...*
```

Sample input 1:

```
11
```

Sample output 1:

```
*....*....*
.*...*...*
..*..*..*
...*.*.*
....***
.....*
.....*
.....*
.....*
.....*
.....*
```

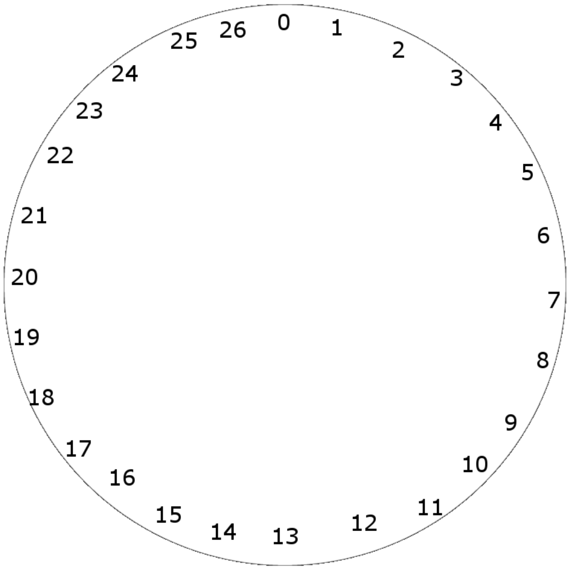
Clocks of Piet

The hands of a clock can be represented by the degree of the angle between then and the zero point, 12. For example if the minute hand was located at 7 minutes past the hour we could say that the angle between it and 12 is 42 degrees. If the minute hand were at 45 minutes past the hour the angle between it and 12 would be 270 degrees.

The planet Piet has a day which is 27 hours long and each hour is comprised of 81 minutes. Your program must print the interior angle between the "hour" hand and the "minute" hand. Only the first 4 decimal digits of your output will count, you may output more digits but it is not necessary.

Tips:

- Keep in mind that at 1:30 on an earth clock the hour hand is halfway between the 1 and the 2.



Example 0: Input:

```
12 45
```

Example 0: Output:

```
32.5925
```

Example 1: Input:

```
26 72
```

Example 1: Output:

```
38.5185
```

Example 2: Input:

```
26 5
```

Example 2: Output:

```
34.7325
```

Camel Case

Normal programming style in Java dictates that method names made from multiple words in the English language should all be written in “camel case”: all the letters in the name should be lowercase, but if a method name is made up of more than one word in the English language, the first letter of each word after the first is capitalized. For example hereIsASentenceInCamelCase.

For this problem, you will be given 1) the number of words in a dictionary, 2) a list of dictionary words, all lowercase and one word per line, and 3) a Java method name to check. You can assume that no dictionary word is a prefix of another word (i.e. if ‘big’ is in the dictionary, then ‘bigger’ won’t be)

Your job is to determine if the method name is composed exclusively of words in the dictionary, and if it has been correctly camelCased. If it is both, print “it is well written” and if not, print “it is not well written.”

```
Sample input 0:            Sample input 1:      Sample input 2:      Sample input 3:
5                          4                    3                    2
a                          a                    a                    frz
big                        big                  big                  pqz
red                        red                  red                  pqz
dog                        dog                  bigred
barks                      bigRedBarksA
bigRedBarksA

Sample output 0:           Sample output 1:      Sample output 2:     Sample output 3:
it is well written         it is not well written it is not well written      it is well
written
```

Identifying Identifiers

In C and other programming languages, names of variables, methods, classes, etc. are called *identifiers*. C has strict rules for naming identifiers. The first two rules are as follows:

1. The name of an identifier consists of uppercase letters, lowercase letters, digits and underscores.
2. The first character of the identifier must be an uppercase letter, a lowercase letter, or an underscore.

For this problem, you will be given an identifier to check, and the program should output whether the input is a valid identifier.

Sample input 0:

```
theVar
```

Sample output 0:

```
theVar is a valid identifier
```

Sample input 1:

```
23thisIsIt
```

Sample output 1:

```
23thisIsIt is not a valid identifier
```

Sample input 2:

```
printF
```

Sample output 2:

```
printF is a valid identifier
```

Monotonic sequence

A monotonic sequence is a sequence that has elements that are either non-increasing (each number is smaller or equal than the previous) or non-decreasing (each number is larger or equal than the previous). You will be given a sequence of seven integers. Your program must print if the sequence is monotonic.

Sample input 0:
```
9 6 5 1 -2 -2 -7
```
Sample output 0:
```
monotonic
```

Sample input 1:
```
0 0 0 0 1 98 99
```
Sample output 1:
```
monotonic
```

Sample input 2:
```
0 1 1 0 0 0 0
```
Sample output 2:
```
not monotonic
```

Binary powers

It is very useful for computer science students to memorize all the values of 2^n and to be able to quickly recognize if an integer is a power of 2. Your program will be given a single positive integer less than two billion. It must print if that number is a power of two.

Sample input 0:
```
      256
```
Sample output 0:
```
      It is a power of 2.
```

Sample input 1:
```
      257
```
Sample output 1:
```
      It is not a power of 2.
```

Space Travel Corporation

The new Space Travel Corp has hired a team of cracker jack programmers like yourselves to diagnose problems in the space traveling routes. They would like to know if it’s possible to get from any station to any other station using known routes. Given a set of stations and the routes between them show whether or not it is possible to get from any station to any other station.

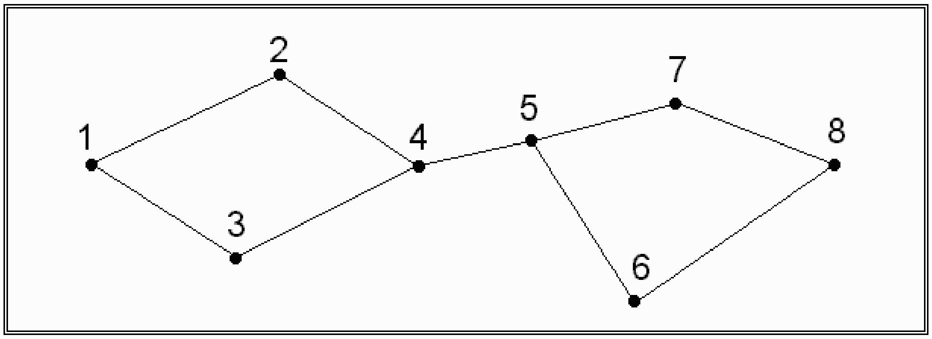


Fig. In the example network of routes to the left it is possible to get from one to any other. Sometimes by moving through another station. 1 is connected to 4 by 3 and 2.

Input will begin with a number which tells you how many connections will be given on subsequent lines. The remaining input will be pairs of station names telling you that there is a rout between them. Each station name is an integer. Each pair will be separated with a space. Ex. 1 2

Output should be “Not enough connections” if it is not possible to reach any station from any other station, alternatively, if it is possible then the output should be “Make it so”.

Example input 0:

```
7
1 2
2 3
3 1
4 5
5 6
6 4
```

Output:

```
Not enough connections
```

Example 1:

```
5
1 2
2 3
3 4
4 1
4 2
```

Output:

```
Make it so
```

Longest repeated substring

Given a string, find the longest substring that is repeated. If there is a tie for the longest repeated substring, you may print any of the longest repeated substrings. You may assume that all the letters in the input are lower case and that the input will always contain some substring that is repeated, even if it’s just a single character.

Sample input 0:

```
there is a dog; here is a door
```

Sample output 0:

```
here is a do
```

Sample input 1:

```
alas
```

Sample output 1:

```
a
```