**Java program:** Prob06.java

**Input File:** Prob06.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

In general, a palindrome is a word, phrase, number, or other sequence of characters that can be read the same way in either direction. The following are all examples of palindromes:

* 1234321
* Never odd or even
* Level
* Too bad – I hid a boot

Your task is to write a program that will check to see if a line of input is a palindrome or not. Any character that is not a letter or a number can be ignored, and case does not matter.

**Program Input**

The file Prob06.in.txt will contain any number of lines of text. Each line should be treated as a new possible palindrome.

**Example Input:**

Test

12321

Lisa Bonet ate no basil

Geese migrate south

**Program Output**

For each line of input, your program should simply state whether or not the line is a palindrome by outputting either "yes" or "no".

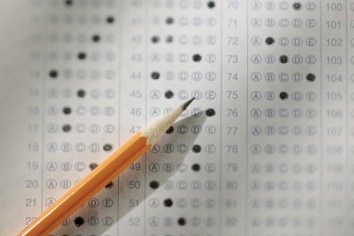
**Example Output:**

no

yes

yes

no

**Java program:** Prob07.java

**Input File:** Prob07.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Scantrons! You know, sheets of paper used to mark your answers on multiple choice tests. You can select A, B, C, D, or E by filling in that little bubble. In this exercise, you will be writing a program to grade students’ scantron papers so the teacher doesn’t have to do it manually.

**Program Input**

The file Prob07.in.txt will contain three sections:

1. The first line of the file will be an integer number telling you how many points each question is worth.
2. The next set of lines will be the key to the test. Blank columns are indicated by an "O", and the bubbled in column is indicated by an equal sign. Columns are separated by a single space. Multiple answers may be bubbled in.
3. The student response section will consist of any number of students’ answers. Each student section will contain the following:
   1. The first line of each student’s section will be in the format "STUDENT #", where the pound sign will be an integer number indicating the student’s ID number. The word student and the ID number will be separated by a single space.
   2. The next set of lines will be the student’s responses to the questions on the test. Questions are answered in order, and only an exact match to the key will receive credit for the question. There is no partial credit.

**Example Input:**

10

O O O O =

O = O O O

O O O O =

= O O O O

= O O O O

O = O O O

O O = O O

O O O = O

O = O O O

O O O O =

STUDENT 12

O = O O =

O = O O O

O O O O =

= O O O O

= O O O O

O = O O O

O O = O O

O O O = O

O = O O O

O O O O =

STUDENT 13

O O O O =

O = O O O

O O O O =

= O O O O

O O O = O

O = O O O

O O = O O

O O O = O

O = O O O

O = O O O

**Program Output**

Your program should output each student’s id number along with their grade in the following format:

STUDENT #: Grade

**Example Output:**

STUDENT 12: 90

STUDENT 13: 80

**Java program:** Prob08.java

**Input File:** Prob08.in.txt

**Output:** Your output needs to be directed to stdout (i.e., using System.out.println())

**Introduction**

Sometimes it’s just nice to take words and make pictures out of them. Your task is to take a phrase and print it out to look like a square or a rectangle.

**Program Input**

The file Prob08.in.txt will contain a list of phrases, one per line. There will be no punctuation or other non-letter characters in each phrase – only words separated by spaces.

**Example Input:**

TIME FLIES LIKE AN ARROW

Squares are fun

**Program Output**

Your program should print out each phrase’s square or rectangle with a blank line in between each to separate the phrases. Use the following rules for determining how to print your squares and rectangles:

* To prevent the end of the phrase from touching the beginning, you should insert a space at the end of each phrase if there is not one there already. Add the minimum possible additional spaces as necessary to fill out your square or rectangle.
* All spaces in the phrase (and added at the end) should be represented in your output by periods.
* Arrange the resulting text in a rectangle/square starting at the upper left corner and proceeding in a clockwise direction. In the event that your phrase forms a rectangle instead of a square, the width of the rectangle must be one greater than the height.

**Example Output:**

TIME.FLI

. E

. S

W .

O L

R I

RA.NA.EK

Squar

. e

n s

u .

f.era