**Java program:** Prob01.java

**Input File:** Prob01.in.txt

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

**Introduction**

When I say CODE, you say QUEST!

CODE……QUEST!

CODE……QUEST!

We are very enthusiastic about this year’s CodeQuest competition. We would like you to join in the enthusiasm by writing a program which exclaims CODE for an input number which is a multiple of 3, QUEST for an input number which is a multiple of 7 and CODEQUESTfor an input number which is a multiple of both 3 and 7.

**Program Input**

The file Prob01.in.txt will contain a list of numbers, one per line.

**Example Input:**

1

3

7

5

9

10

11

21

42

52

**Program Output**

For each input number:

* If the number is a multiple of 3, output **CODE**.
* If the number is a multiple of 7, output **QUEST**.
* If the number is a multiple of both 3 and 7, output **CODEQUEST**.
* Otherwise, just output the number which was read in.

**Example Output:**

1

CODE

QUEST

5

CODE

10

11

CODEQUEST

CODEQUEST

52

**Java program:** Prob02.java

**Input File:** Prob02.in.txt

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

**Introduction**

A deck of cards contains 52 cards. Some are red, some are black, and there are 4 different suits. You will be dealt hands of cards from a standard deck. Group each hand by color, then by suit, and then by face value. Your output should list the amount of cards in each grouping.

**Program Input**

The file Prob02.in.txt will contain the following:

* A hand of cards will appear on one line. Cards will be separated by spaces.
* Each hand will be on a separate line.
* Each hand will have a random number of cards.
* The cards will be listed by face value then suit (i.e. 5C would be the 5 of Clubs).
* There are 4 suits in a deck of playing cards: Clubs, Diamonds, Hearts, and Spades.
* Spades and Clubs are always BLACK.Hearts and Diamonds are always RED.
* The letters J, Q, K, and A will represent Jack, Queen, King, and Ace respectively.
* No Jokers will be dealt.

**Example Input:**

9H JS AC 3H 10S 9C 5H 7C 4S QS 2D 2S 8H 9D 3D 7H 7S AS 10H

AS JC 7H 8D KH JC 5C 9H 10D 2S 7S 6H

JH KD 10D 10H 2H AH 8D 7H 5H 4D 9H 3D KH

**Program Output**

Your output should contain the groupings of cards in each hand. The total number of cards for each group should be printed followed by a dash (-) and then the name of the group. If a color or a suit is not in the hand, a value of 0 should be printed for the group. Face values that are not present should be omitted from the output. For the color lines, always use the words “RED” and “BLACK” (i.e. don’t put a trailing S on either category). For the suit and face value lines, use plurals as appropriate.

Separate each hand by a title of “HAND #” in the order that it was given in the input file. Print the groups in the following order for each hand:

* COLOR (in this order: RED, BLACK)
* SUIT (in suit order: CLUB, DIAMOND, HEART, SPADE)
* FACE VALUE (face value should be printed in ascending order – 2 is low, Ace is high.

**Example Output:**

HAND 1

9-RED

10-BLACK

3-CLUBS

3-DIAMONDS

6-HEARTS

7-SPADES

2-2 cards

2-3 cards

1-4 card

1-5 card

3-7 cards

1-8 card

3-9 cards

2-10 cards

1-Jack

1-Queen

2-Aces

HAND 2

6-RED

6-BLACK

3-CLUBS

2-DIAMONDS

4-HEARTS

3-SPADES

1-2 card

1-5 card

1-6 card

2-7 cards

1-8 card

1-9 card

1-10 card

2-Jacks

1-King

1-Ace

HAND 3

13-RED

0-BLACK

0-CLUBS

5-DIAMONDS

8-HEARTS

0-SPADES

1-2 card

1-3 card

1-4 card

1-5 card

1-7 card

1-8 card

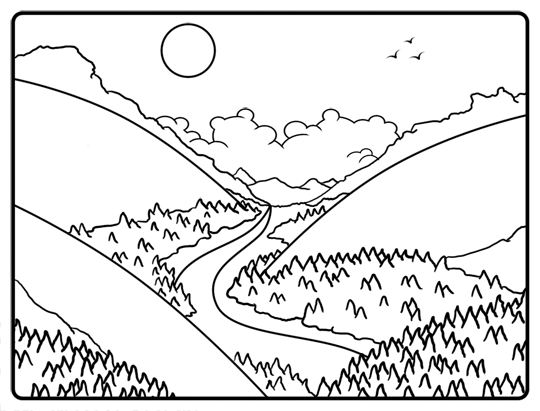
1-9 card

2-10 cards

1-Jack

2-Kings

1-Ace

**Java program:** Prob03.java

**Input File:** Prob03.in.txt

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

**Introduction**

Sorting is one of the most common things you are asked to do as a programmer. In fact, higher level languages and third party libraries make it easier to do things like sort numbers without even knowing what’s happening behind the scenes. We say “no more”! Today, you will implement the valley sort algorithm on lists of numbers.

What? You’ve never heard of the valley sort? That’s because we made it up. At least we think we did – we Googled it and didn’t find anything similar. The valley sort goes something like this:

The biggest number should wind up being first in the list after sorting.

The second biggest number should wind up being last in the list after sorting.

The third biggest number should wind up being second in the list after sorting.

The fourth biggest number should wind up being second from the last in the list after sorting.

And so on. Check the example input and output for a couple of visual examples. We think they will make sense.

**Program Input**

The file Prob03.in.txt will contain several lines of integers ranging from 0 to Integer.MAX\_VALUE. Each line is a list of numbers separated by spaces. Each line should be run through your valley sort algorithm, and the results printed out.

**Example Input:**

5 4 3 2 1

1 2 3 4 5 6 7 8 9 10

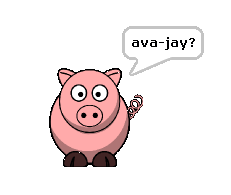
**Program Output**

Your program should print the valley sorted lists of numbers, one list per line, in the order they were read in. Separate your numbers by a single space.

**Example Output:**

5 3 1 2 4

10 8 6 4 2 1 3 5 7 9

**Java program:** Prob04.java

**Input File:** Prob04.in.txt

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

**Introduction**

Pig Latin is a mysterious and often overlooked way of communicating. We want to make sure that the art of speaking Pig Latin is not lost. To help with this worthwhile task you will need to write an English to Pig Latin translator.

For words that begin with a single consonant: take the consonant off the front of the word and add it to the end of the word. Then add “ay” to the very end. Here are some examples:

cat = atcay dog = ogday simply = implysay noise = oisenay

For words that began with multiple consonants: take the group of consonants off the front of the word and add them to the end of the word. Then add “ay” to the very end. Here are some examples:

scratch = atchscray thick = ickthay flight = ightflay grime = imegray

For words that begin with a vowel, just add “yay” at the end. For example:

is = isyay apple = appleyay under = underyay octopus = octopusyay

For the sake of simplicity, we will consider the letter y to always be a consonant. The only exception to these rules is the letter q. Since the letter q needs the letter u, if you find a “qu” sequence in a word, treat that sequence as a single consonant character. You will see an example of that in the example output below.

For words that do not contain a vowel, simply keep the word the same.

**Program Input**

The file Prob04.in.txt will contain phrases to be translated, one per line, allin lower case.

**Example Input:**

piglatin rocks

code quest is the best

java rules

**Program Output**

You should output the phrases translated to Pig Latin in the order they were in the input file.

**Example Output:**

igpayatinlayocksray

odecayestquayisyayethayestbay

avajayulesray