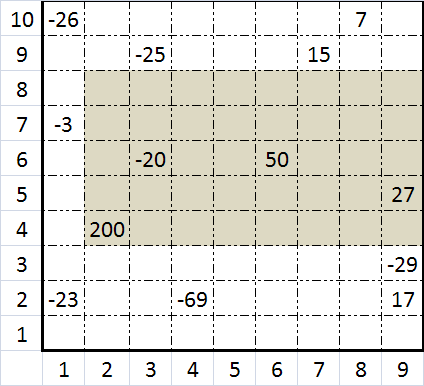
**Java program:** Prob13.java

**Input File:** Prob13.in.txt

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

**Introduction**

Two friends have been using instant messenger and have also been watching too many spy movies. They are afraid that people will be able to listen in on their conversation, so they use steganography to hide what they are talking about. The friends send each other YouTube links that follow this pattern:

https://www.youtube.com/watch?v=UYU8Q3

This looks like any normal YouTube link to the untrained eye, but the text after the “v=” is actually an encoded message. Your job is to write a program that will decode a conversation encoded using these YouTube links. The encoding is done using the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Value | Encoding | Value | Encoding |
| 0 | A | **8** | Y |
| 1 | B | **9** | 3 |
| 2 | X | **10** | R |
| 3 | 1 | **11** | G |
| 4 | 5 | **12** | K |
| 5 | 8 | **13** | 4 |
| 6 | U | **14** | E |
| 7 | Q | **15** | 7 |

Every character in normal text is split into two encoded characters as follows:

1. Find the ASCII value of the character (for example, the letter h has an ASCII value of 104, or 01101000).
2. Use the integer value of the first 4 bits to get the first encoded character (for h, the first 4 bits are 0110, or 6, which maps to U from the table).
3. Do the same for the second 4 bits (for h, the second 4 bits are 1000, or 8, which maps to Y from the table).
4. Put these encoded characters together and repeat the process for each character in the string you are encoding (so h would map to UY).

Your program should reverse this process to find the actual text from the encoded link parameters.

**Program Input**

The file Prob13.in.txt will containa conversation between two or more people using the fake links containing the encodings. Their conversation will only use alphanumeric characters, and special characters such as ‘=’ will not be used in the conversation.

**Example Input:**

Alice: https://www.youtube.com/watch?v=UYU8Q3

Bob: https://www.youtube.com/watch?v=QQUYUBQ5XAU3Q1XAU3Q5

Alice: https://www.youtube.com/watch?v=U1UBUEXAQQU8XAU4U8U8Q5

Bob: https://www.youtube.com/watch?v=5E57

Bob: https://www.youtube.com/watch?v=53XAU5U7UEQ5XAUGUEU7QQXAQ3U7Q8

**Program Output**

Your program’s output should display the conversation in the same format as the input but with the decoded values instead of the links.

**Example Output:**

Alice: hey

Bob: what is it

Alice: can we meet

Bob: NO

Bob: I dont know you

**Java program:** Prob14.java

**Input File:** Prob14.in.txt

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

**Introduction**

Word search puzzles have provided a challenging game for all ages since they were first introduced over 40 years ago. They require a keen eye and plenty of patience to complete. Seemingly random letters are aligned in a square or rectangular grid. Generally, they are kept to 30x30 or less but they can be of any size. Given a list of words, the solver must find every listed word in the grid. The words can be found in the following different directions: across, up, down, diagonally, and reversed in any direction. Your task is to create a program that will find the words in the grid.

**Program Input**

The file Prob14.in.txt will containa grid of uppercase letters. The grid can be up to any size and will not necessarily be square. The line above the grid will have the string “#PUZZLE” without the quotes. Following the grid there will be a list of words, each on a separate line. The line above the beginning word in the list will have the string “#WORDS” without the quotes. Remember that the first row and first column of the grid start at 0, not 1. In the following example, the first letter “J” is at location [0,0], not [1,1].

**Example Input:**

#PUZZLE

JOYQFT

KIRELO

BHJASP

MITEQA

UJHGJI

GTKRUO

#WORDS

THE

JOY

TOP

OUR

**Program Output**

The output of your program should list the words, sorted alphabetically, each followed by the beginning letter’s location in the grid in [row,column] coordinate format. Make sure to place a single space between the word and its location. Each word / location pair should be on its own line. All words should be found. If there are two or more instances of a word found, report the one with the lowest column value coordinate. If two or more instances have the same x value, report the one with the lowest row value coordinate.

**Example Output:**

JOY [0,0]

OUR [5,5]

THE [5,1]

TOP [0,5]