Problem 1: When I Say Code, You Say Quest!
2 points

CODE QUEST 2014

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 `CODEQUEST` for 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
```

Problem 1: When I Say Code, You Say Quest!
2 points

CODE QUEST 2014

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
```

Problem 2: Deal the Cards
3 points

CODE QUEST 2014

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.

Problem 2: Deal the Cards
3 points

CODE QUEST 2014

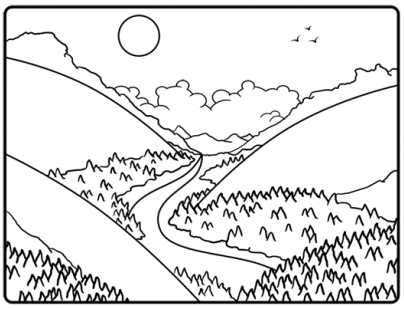
Example Output:

```
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
```

Problem 3: Valley Sort
4 points

CODE QUEST 2014

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
```

Problem 4: Pig Latin
5 points

CODE QUEST 2014

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.

Problem 4: Pig Latin
5 points

CODE QUEST 2014

Program Input

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

Example Input:

```
pig latin 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:

```
igpay atinlay ocksray
odecay estquay isyay ethay estbay
avajay ulesray
```

Problem 5: Book Worm
6 points

CODE QUEST 2014

Java program: `Prob05.java`
Input File: `Prob05.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

The invention of e-reading technology on computers, cell phones, tablets, and e-readers is revolutionizing the book industry. In years past, there were a handful of publishers capable of bringing a book to publication. Now, almost anyone with a computer can become an independent publishing house. As the number of publishers grow, so too will the number of books. This is beneficial to authors and consumers, but can be problematic for standards organizations that are charged with maintaining order among industries.

A book's ISBN (International Standard Book Number) is a way to uniquely identify a work while simultaneously providing information about that work. ISBNs currently have 13 digits broken down into the following sections:

1. A 3 digit prefix assigned by the GS1 standards organization (currently 978 or 979, but they could be anything).
2. A registration group number that varies from 1 to 5 digits. This number identifies the region or language of the work.
3. A registrant element, which identifies a specific publisher.
4. A publication element, which identifies a work published by the publisher.
5. A single check digit (used to verify the legitimacy of the ISBN).

The check digit is assigned in such a way that taking the sum of all 13 numbers, each multiplied by a factor alternating between 1 and 3, is a factor of 10. For example, for ISBN 978-0-306-40615-7:

```
 9*1 + 7*3 + 8*1 + 0*3 + 3*1 + 0*3 + 6*1 + 4*3 + 0*1 + 6*3 + 1*1 + 5*3 + 7*1
=  9 +  21 +   8 +   0 +   3 +   0 +   6 +  12 +   0 +  18 +   1 +  15 +   7
= 100
```

Since 100 is a factor of 10, this is a valid ISBN number. There is no standard format for an ISBN. ISBNs can contain dashes or other separating characters (spaces, underscores, etc.) that can make them more readable by humans, or they can just be 13 digits in a row. Your job is to validate a list of ISBN candidate numbers.

Problem 5: Book Worm
6 points

CODE QUEST 2014

Program Input

The file `Prob05.in.txt` will contain a list of ISBN numbers, one per line, which you must validate.

Example Input:

```
978-0-306-40615-7
9788175257665
1234567890123
```

Program Output

For each ISBN number your program should output one of the following:

- If the ISBN is valid, output the word `VALID`
- If the ISBN is invalid, output the value that the check bit should have been to make the ISBN valid

Example Output:

```
VALID
VALID
8
```

Problem 6: Product of a Grid
7 points

CODE QUEST 2014

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

Everyone played connect four as a kid, right? Well, you might call this problem "multiply four". Given a square grid of non-negative integers, you must find the greatest product of any four adjacent numbers in any direction (up, down, diagonally, left, right). All numbers in the grid will be less than 100.

Program Input

The file `Prob06.in.txt` will contain a square grid of integers. Numbers on the same line will be separated by a single space.

Example Input:

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

Program Output

Your program should print out the greatest product of any four adjacent numbers in the grid using the format shown below. The solution is highlighted above for clarity.

Example Output:

```
Greatest product: 70600674
```

Problem 7: Morse Code
8 points

CODE QUEST 2014

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

Everything in a computer boils down to ones and zeroes – things we call bits. More accurately, we think of a bit represented by a 1 to be considered "on", and that is measured by the presence of an electric charge. We think of a bit represented by a 0 to be considered "off", and that is measured by the lack of an electric charge. Today we have computers that have billions of bits that we can put together to form useful information – information that can be readily consumed by untrained users (like text on a screen). But what if there were only a single bit?

Back in 1836, Samuel Morse needed a way to transmit messages using only electric current. The problem: how to represent text using only an "on" or an "off" signal at a single node. The answer: the length of time that the bit was on or off would have to mean something. The product: Morse code! Here is how Morse code works:

- The building blocks of Morse code are the "dot" and the "dash". A dot is an electric signal with a length of one time unit. A dash is an electric signal with the length of three time units.
- The space between dots and dashes within a single letter has a length of one time unit.
- The space between letters in the same word has a length of three time units.
- The space between two words has a length of seven time units.

Your task has two parts: first, read in some text and turn it into Morse code. Second, read in some Morse code and decode it into text. The last page of this problem has a translation table for you to use when coding your solution.

Program Input

The file `Prob07.in.txt` will contain the following:

- Some number of lines of text that you are to read in and change to Morse code. Line breaks should be preserved, so your output should have the same number of lines that the input text has. There will be only text – no numbers, and no special characters or punctuation.
- A line reading END OF TRANSMISSION which will serve as the break point between the text and the Morse code that you are to interpret.
- Some number of lines of Morse code that you are to read in and change to plain text. Again, line breaks should be preserved.
- A line reading END OF TRANSMISSION which will denote the end of the file

Problem 7: Morse Code
8 points

CODE QUEST 2014

Example Input:

```
It is I
Code Quest
END OF TRANSMISSION
===_=_===_=___===_===_===___===_=_=___=_______===_===_=_===___=_=_===_
__=___=_=_=___===_______=_===_=___=_=_===___=_===_=_=___=___=_=_
=_=_____=_=_=_=___===_===_===___=_===_===_=___=_____===_=_===_===_
_===_===_===___=_=_===_____===_===_=___=_=_===___===_=_===_===___=_
=_=_____=_===_=_=___=_=___===_=_===___=_____=_===_===_=___=_=___==
=_===_=_=___===_===_=_=___=_===
END OF TRANSMISSION
```

Program Output

Because Morse code deals only with the presence or absence of an electrical current, you will use the following when translating between text and Morse code:

- The presence of an electrical current will be represented by an equal sign (=).
- The absence of an electrical current will be represented by an underscore (_).

Your output should mimic the input file by having the following four elements:

- The Morse Code representation of the text from the input file
- A line containing the phrase END OF TRANSMISSION
- The text representation of the Morse Code from the input file (all in lowercase)
- A line containing the phrase END OF TRANSMISSION

Make sure you take a look at the electronic copy of this input and output – the Morse Code sections are too long to fit on one printed line, so they are wrapping here in the printed packet.

Example Output:

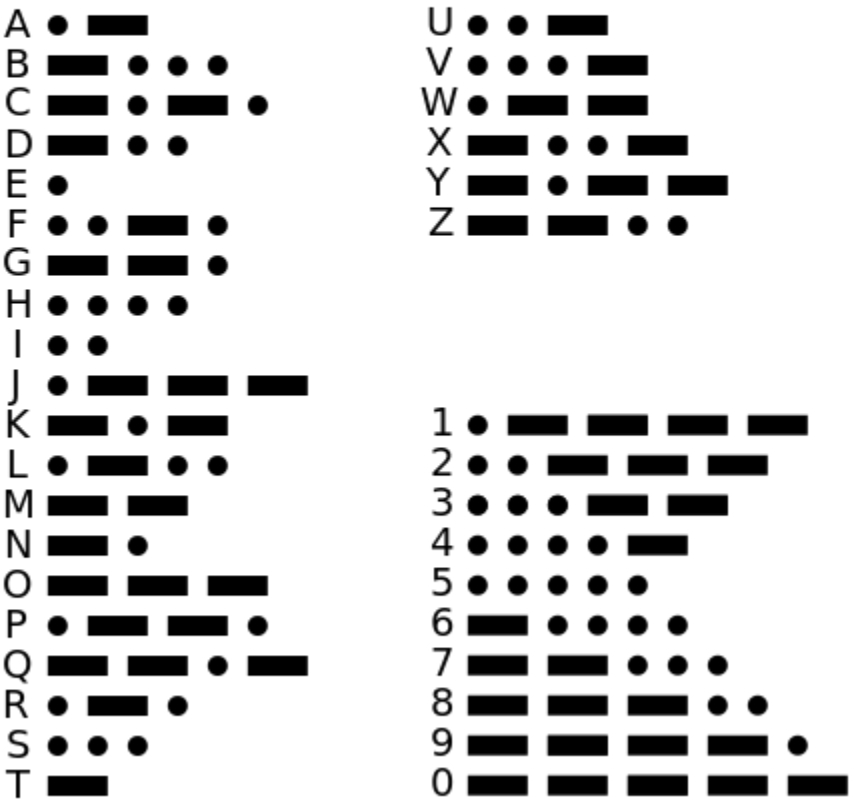
```
=_=___===_____=_=___=_=_=_____=_=
===_=_===_=___===_===_===___===_=_=___=_____===_===_=_===___=_=_===_
__=___=_=_=___===
END OF TRANSMISSION
code quest rules
i hope you guys like pizza
END OF TRANSMISSION
```

Problem 7: Morse Code
8 points

CODE QUEST 2014

International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.



Problem 8: Rectangle Art
10 points

CODE QUEST 2014

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

Have you ever seen a spray paint artist? Their art depends on layering – they spray lots of colors on top of each other and then scrape paint away to make beautiful scenes. In this problem, you will deal with layered rectangles in a workspace grid. Your art will involve these rectangles layered on top of each other.

Lockheed's stealth technology has influenced the world of rectangle art by providing rectangle artists with a mysterious material that turns clear when there are even numbers of layers of that material touching each other. So, any square in the grid that is covered by an odd number of rectangles appears covered, while squares in the grid with an even number of rectangles covering a space appear to be uncovered.

Your task is to write a program that will compute what a workspace will look like after placing these special rectangles down. All your inputs will be given in Cartesian format.

Program Input

The file `Prob08.in.txt` will contain the following:

- The first line of the file will be the workspace area in X,Y format – meaning that the workspace area is X units wide and Y units tall. The origin will be in the bottom left corner of the workspace, so all coordinates will be positive. The numbers will be separated by a single comma.
- The rest of the lines in the file will describe the location of the rectangles in the workspace. Each line will contain coordinates for two points that will be the diagonal corners of the rectangle. The x and y coordinates for each point will be separated by a single comma, and there will be a single space between the two coordinates.

Problem 8: Rectangle Art
10 points

CODE QUEST 2014

Example Input:

```
20,20
0,10 10,20
20,9 10,20
0,0 20,9
4,13 6,15
14,15 16,13
3,6 4,8
4,5 5,7
5,4 6,6
6,4 7,5
6,4 14,3
13,4 15,5
14,5 16,6
15,6 17,7
17,8 16,7
0,9 10,10
0,20, 20,0
0,0 20,1
0,19, 20,20
0,1 1,19
19,1 20,19
```

Program Output

Your output should produce a grid where a space means a spot looks clear and an asterisk means a spot looks covered.

Example Output:

```
********************
*                  *
*                  *
*                  *
*                  *
*    **        **   *
*    **        **   *
*                  *
*                  *
*                  *
*                  *
*                  *
*                  *
*  *            *  *
*  **          **  *
*   **        **   *
*    **      **    *
*     ********     *
*                  *
*                  *
********************
```

Problem 9: Check Please!
12 points

CODE QUEST 2014

Java program: `Prob09.java`
Input File: `Prob09.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

OK, OK, we know everyone pays their bills online these days but there are still some occasions where a check is called for. To help with these few occasions, write a program that converts dollar amounts from their numeric form into their English equivalent, as would be seen on a check.

Guidelines:

1. Not all input numbers will have a decimal. Numbers with no decimal should be assumed to have 0 Cents.
2. There will be no more than 2 numbers after the decimal point.
3. Capitalize each English word except the word "and".
4. All inputs will be less than one million.
5. You should use the singular for Dollar and Cent when appropriate.

Program Input

The file `Prob09.in.txt` will contain a list of numbers, one per line.

Example Input:

```
143
2.34
1.01
1234.56
```

Program Output

Your program should output the English version of each number in the input file in the order it was encountered in the file.

Example Output:

```
One Hundred Forty Three Dollars and 0 Cents
Two Dollars and 34 Cents
One Dollar and 1 Cent
One Thousand Two Hundred Thirty Four Dollars and 56 Cents
```

Problem 9: Check Please!
12 points

CODE QUEST 2014

Reference

For your reference, here are the first 100 integers written in English:

Zero	Twenty Five	Fifty	Seventy Five
One	Twenty Six	Fifty One	Seventy Six
Two	Twenty Seven	Fifty Two	Seventy Seven
Three	Twenty Eight	Fifty Three	Seventy Eight
Four	Twenty Nine	Fifty Four	Seventy Nine
Five	Thirty	Fifty Five	Eighty
Six	Thirty One	Fifty Six	Eighty One
Seven	Thirty Two	Fifty Seven	Eighty Two
Eight	Thirty Three	Fifty Eight	Eighty Three
Nine	Thirty Four	Fifty Nine	Eighty Four
Ten	Thirty Five	Sixty	Eighty Five
Eleven	Thirty Six	Sixty One	Eighty Six
Twelve	Thirty Seven	Sixty Two	Eighty Seven
Thirteen	Thirty Eight	Sixty Three	Eighty Eight
Fourteen	Thirty Nine	Sixty Four	Eighty Nine
Fifteen	Forty	Sixty Five	Ninety
Sixteen	Forty One	Sixty Six	Ninety One
Seventeen	Forty Two	Sixty Seven	Ninety Two
Eighteen	Forty Three	Sixty Eight	Ninety Three
Nineteen	Forty Four	Sixty Nine	Ninety Four
Twenty	Forty Five	Seventy	Ninety Five
Twenty One	Forty Six	Seventy One	Ninety Six
Twenty Two	Forty Seven	Seventy Two	Ninety Seven
Twenty Three	Forty Eight	Seventy Three	Ninety Eight
Twenty Four	Forty Nine	Seventy Four	Ninety Nine

Problem 10: All Aboard!
14 points

CODE QUEST 2014

Java program: `Prob10.java`
Input File: `Prob10.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Since their invention over 200 years ago, trains have provided a safe and efficient method of transportation. Passenger trains carry hundreds of travelers from station to station, requiring only track to be laid between them. In the beginning, train stations had dedicated tracks which linked them. This proved to be a waste of infrastructure as much track was going unused the majority of the time. Railroad switches were designed to solve this problem. Switches allow trains to quickly cross to an adjacent track. In this manner, trains were no longer bound to the track they began their journey on. Your task will be to determine, for each starting station, what the end station is given a layout of interconnected tracks.

Program Input

The file `Prob10.in.txt` will contain a set of train station names, tracks, barriers, and switches.

- Train tracks are denoted by an equal sign (=). Each track has a start station and an end station. Trains move from left to right.
- Station names will always be three characters and will appear at the beginning and end of a track. Station names are not necessarily unique.
- Between two adjacent tracks will be barriers and switches. Barriers are denoted by a minus sign (-), and switches are denoted by a vertical bar (|). When a train reaches a switch, it will always crossover to the neighboring track. For example:

```
AAA======================BBB
   ------|---------|---
AAB====================CCC
   --------|---------|----------
AAC============================DDD
```

A train leaving station AAA would cross to the middle track, then to the bottom track, and then back to the middle track to arrive at station CCC.

- Switches will never connect directly to a station – they can only connect two track pieces.
- If the situation occurs where it's possible for a train to switch to either of the two neighboring tracks, the train will always move in the "upward" direction. Also, switches can be used to cross multiple tracks if they line up correctly. For example:

In the example on the left, a train leaving station AAB would arrive at the switches in the middle and would choose to move up instead of down. The train would end up at station BBB. In the example on the right, a train leaving station AAA would cross over the middle track because the switches are aligned. The train would arrive at station BBC.

Example Input:

```
AAA=============================================================GGG
   -|-----|------|--------|-----------|--------------------
BBB=======================================HHH
   --------|----------------|-----------|----------
CCC=================================================III
   -----|--------------|--------------------|-----------------
DDD=========================================================JJJ
   -------|-------------|-----------------------------------
EEE============================KKK
   ------|----------------|---------------
FFF=======================================LLL
```

Program Output

Your program should list out the starting stations in the order they were encountered in the input file. For each starting station, list the ending station for a train leaving from that starting station in the following form:

```
Start: StartStation, End: EndStation
```

Example Output:

```
Start: AAA, End: JJJ
Start: BBB, End: LLL
Start: CCC, End: III
Start: DDD, End: HHH
Start: EEE, End: KKK
Start: FFF, End: GGG
```

Problem 11: Spiral Text
16 points

CODE QUEST 2014

Java program: `Prob11.java`
Input File: `Prob11.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

If you've ever looked at a word search puzzle and thought "hey, I wonder if there's some hidden message in there somewhere?" then this problem is for you. Your task is to take some text and create a counter clockwise spiral out of it.

Program Input

The file `Prob11.in.txt` will contain some number of lines of text. Your program should read in the text and print it out in a counter clockwise spiral. Here are the parameters for creating the spiral:

- The spiral should be contained within a square, and the square should be as small as possible.
- The first character of the text should go in the exact center of the square.
- The second character of text should go to the left of the first character.
- The spiral of text should go counter clockwise from there.
- Insert spaces as necessary to separate words from different lines (watch out for input lines with leading and/or trailing spaces).
- When you reach the end of the text, fill the remainder of the spiral with spaces.

Example Input:

```
You should
turn this
text into
 a spiral.
```

Note: take a look at the input file for this – there are spaces in the file that you can't see here.

Program Output

Your program should print the text in spiral form as described above. Again, take a look at the example file provided. The extra spaces are not readily apparent here.

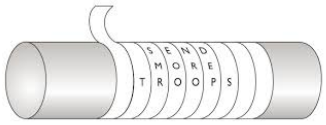
Example Output:

```
 txet
idluos
n oYhi.
ttu shl
ourn ta
 a spir
```

Problem 12: Caesar-Scytale
18 points

CODE QUEST 2014

Java program: `Prob12.java`
Input File: `Prob12.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)



Introduction

The Caesar and Syctale ciphers are very old ciphers used by the ancient Greeks to encrypt secret messages. In this problem you will be decrypting a message that has been encoded using both methods.

The Caesar Cipher

The Caesar cipher is a form of a substitution cipher because it substitutes letters for other letters. A Caesar cipher just shifts letters however many positions to the left or the right. For example:

```
LOCKHEED --> right shift 3 --> ORFNKHHG
```

Letters are shifted as if we were using an alphabet ring. That is, shifting Z to the right produces an A, and so forth. So, the string "Zebra" shifted 1 position to the right would be "Afcsb". Note that capitalization is retained.

The Scytale Cipher

The Scytale cipher is a form of transposition in that it rearranges letters into different positions. In a Scytale cipher, you would wrap a very long piece of paper around a stick and write your message going across. The message would be unreadable when unwrapped. You had to use the same stick to decrypt the message. This is similar to picking a two dimensional array and filling it with the message. For example, the word "LOCKHEED" using a 4x3 array would produce this:

L	O	C
K	H	E
E	D	X
X	X	X

A capital X is used to pad the array if the message doesn't fill the array exactly. To "unravel" the paper, read the array from top to bottom, left to right:

```
LOCKHEED --> LKEXOHDXCEXX
```

Problem 12: Caesar-Scytale
18 points

CODE QUEST 2014

Putting Them Together

For this problem, you must decrypt a message that was encrypted using the Caesar cipher followed by a Scytale. Let's look at the "LOCKHEED" example:

```
LOCKHEED --> Caesar +3 --> ORFNKHHG --> Scytale 4x3 --> ONHXRKGXFHXX
```

There are a couple of things that are important to note:

- If the input string was too short to fill the array exactly, the encrypted Scytale string will have more characters than the original decrypted message. Be careful when printing your output because of this. You can assume that the original message does not end in a capital X.
- You will not know the array size that was used to perform the Scytale cipher, but it is guaranteed that the number of characters in the encoded input file will fit exactly into the array used.
- You will not know the shift value that was used to perform the Caesar cipher, but the original text is guaranteed to have the word "Dear" in it. Shifting the letters in the word "Dear" n times produces no other word in the English language, so it is a sufficient key to know that the decoding has been successful.

Program Input

The file `Prob12.in.txt` will contain some number of lines of encoded text. The file in its entirety should be decoded all at once – each line is not separately encoded. Rather, the lines together form one long string for encoding and decoding purposes.

Example Input:

```
JouyWOjkg
rkgjIacuyzrgxguk
oszoXxeejyreZ
yXJZokzrh
ngX
```

Program Output

Your program should output the decoded message. Your line lengths should match the input file, with the additional padded characters (if any) removed from the end of the file.

Example Output:

```
DearDiary
TodayisCodeQuest
Iwilldomybest
Thatisall
```

Problem 13: Diamond Path
25 points

CODE QUEST 2014

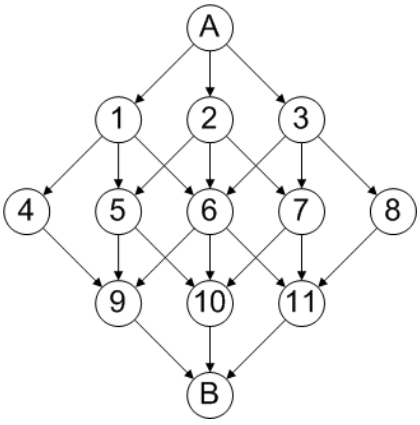
Java program: `Prob13.java`
Input File: `Prob13.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Choices are everywhere. Left or right? Beef or chicken? To be or not to be? These are the questions. But your task involves a different kind of choice – one that will take you from point A to point B.

Choosing a path from point A to point B involves evaluating your choices at each step along the way. During the first half of your journey, the possibilities are many. Specifically, at each point along the way you will have to choose to move in one of three directions.

However, sometimes in life the closer we get to our destination, the fewer choices we have in terms of how to get there. Therefore, for the second half of the journey, your choices will start to narrow. The set of all possible paths will look like a diamond.



Starting at node A, you can choose to go to node 1, 2, or 3. Your choice there determines your next set of choices and so on. Once you hit the middle, your choices narrow if you are on the edge. Being in node 6 means you still have three choices to get to the next level, but being in nodes 5 or 7 reduce your options to 2. Nodes 4 and 8 have zero choice.

Each path will be given a weight representing the difficulty of going down that path. Your job is to determine the path (or paths) of least resistance in getting from point A to point B.

Problem 13: Diamond Path
25 points

CODE QUEST 2014

Program Input

The file `Prob13.in.txt` will contain path weights separated by spaces. All paths will have a weight. Path weights will be grouped based on the nodes above them from left to right. For example, the input file will have all three path weights associated with node 1 before the three path weights associated with node 2. Paths that do not exist will be omitted.

Each line of input will contain the connectors for each level separated by spaces. There will always be a multiple of three connectors per line.

Example Input:

```
1 2 3
2 1 1 1 1 1 1 1 1
0 2 1 1 2 1 2 1 2
1 3 1
```

Program Output

Your program should report two things:

1. The lowest possible path difficulty. Use the format "`Lowest path difficulty: #`".
2. The number of paths that have the lowest possible path difficulty. Use the format "`Number of paths with the lowest difficulty: #`".

Example Output:

```
Lowest path difficulty: 4
Number of paths with the lowest difficulty: 3
```

Problem 14: 3D Bingo
30 points

CODE QUEST 2014

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

If you were here last year then you may remember Mabel, our resident bingo lover. Since you coded a solution to check her bingo cards, she has grown bored of the traditional bingo game. In order to challenge herself again, she started a 3D Bingo parlor to kick it up a notch. Your task is to help her check for bingos. The following information will help you.

Traditional bingo:

A bingo card utilizes the numbers 1 through 75. The five columns of the card are labeled 'B', 'I', 'N', 'G', and 'O' from left to right. The center space is usually marked "Free" or "Free Space", and is considered automatically filled. The range of printed numbers that can appear on the card is normally restricted by column, with the 'B' column only containing numbers between 1 and 15 inclusive, the 'I' column containing only 16 through 30, 'N' containing 31 through 45, 'G' containing 46 through 60, and 'O' containing 61 through 75. The object of the game is to fill 5 consecutive spaces (a whole row, a whole column, or one of the two diagonals) – which is called a bingo. The game ends when a bingo is found on one or more cards.

3D bingo:

3D bingo is very similar to traditional bingo, except that sets of 5 cards can be used to form a “bingo cube”, allowing for a bingo in more ways (straight up and diagonally up). Obviously the free space would cause an automatic bingo, so it has been eliminated in the 3D version.

Mabel really wanted a challenge keeping up with her cards, so she decided that cards could be used in any combination to form a cube. This means that if you buy 4 cards, you have to rely on a single card for a bingo. However, if you buy 5 cards, then you can make a bingo cube. Since the order of the cards does not need to be set before the game starts, there are 120 ways to make the bingo cube out of 5 cards. Buying a sixth card means that you can make 720 different bingo cubes out of your cards. You can see how this game would be much more challenging to keep track of.

Your task is to write a program that will tell Mabel how many possible bingo combinations there are during a 3D bingo game.

Problem 14: 3D Bingo
30 points

CODE QUEST 2014

Program Input

The file `Prob14.in.txt` will contain two sections:

1. The first section will contain the bingo card information. There will be 5 input lines per card. Each line will contain the numbers for one row of a bingo card separated by spaces. The first 5 rows will make up card number 1, the second 5 rows will make up card number 2, and so on. The number of rows will be a multiple of 5 (meaning there are no extra rows to throw away).
2. The second section will be the numbers called during the game. The word PLAY will appear on a line by itself to separate the card information from the game simulation. Following the word PLAY each line will contain a value to be played on your bingo cards (i.e. B3, I27, or B5). Once a bingo is found, the game stops. Any extra input after a bingo is found should be ignored.

Example Input:

```
10 23 36 49 67
13 26 34 50 70
14 20 42 59 71
6 25 35 47 72
7 27 31 57 61
9 24 36 48 65
11 16 41 47 71
1 19 43 56 64
2 17 44 53 62
3 18 42 51 66
12 20 39 59 69
4 22 41 55 61
6 29 33 53 72
14 17 37 58 65
11 16 44 47 66
7 26 31 57 73
1 21 40 55 61
8 29 43 47 62
11 27 35 46 66
6 16 36 52 67
5 16 33 47 62
7 27 35 52 66
11 22 44 51 70
12 19 36 49 72
13 21 41 59 73
PLAY
I28
G46
I16
N43
B3
O67
O62
```

```
O64
G49
G54
B7
N44
O63
B5
G57
B6
I27
N38
G55
B8
B11
O73
```

Program Output

Your program should print the total number of bingos found using the format shown below.

Example Output:

```
Number of bingos: 2
```

Problem 15: Pretty Print
30 points

CODE QUEST 2014

Java program: `Prob15.java`
Input File: `Prob15.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

XML (Extensible Markup Language) is a powerful tool for moving data around because the user is allowed to define tag names. If written properly, XML can be interpreted easily by both humans and machines. However, if written poorly, XML can be a nightmare for a human to read. Your task is to write an XML pretty printer. We'll start with a little XML lesson in case you are unfamiliar.

Every XML document starts with a declaration line that looks something like this:

- `<?xml version="1.0" encoding="UTF-8" ?>`

The declaration section starts with the character sequence `<?xml` and ends with the character sequence `?>`.

After that, the rest of the XML document is made up of XML elements. XML elements are denoted by tags. A tag is a string of characters surrounded by less than (<) and greater than (>) signs. There are three types of tags:

- A start tag starts with a less than sign and ends with a greater than sign marks the beginning of an XML element: `<element>`
- An end tag starts with a less than sign with a forward slash and ends with a greater than sign marks the end of an XML element: `</element>`
- A self-closing tag, which combines the start and end tags into a single tag starts with a less than sign and ends with a forward slash with a greater than sign: `<element />`

XML tags may also contain attributes. Attributes have information that pertains to the XML element they appear in. Attributes appear inside XML tags and are name-value pairs separated by an equal sign, with the value being double quoted. Here is an example of an XML start tag with two attributes:

```
<myTag att1="attribute 1 value" attribute2="att2value">
```

The content of an XML element is what comes between the start tag and the end tag. XML elements can contain plain text content, or they can contain other XML elements. If you are familiar with XML, this problem will not deal with escaped special characters or CDATA.

Problem 15: Pretty Print
30 points

CODE QUEST 2014

Program Input

The file `Prob15.in.txt` will contain an XML file. It will not be pretty, but it will be valid XML.

Example Input:

```
<?xml version="1.0" encoding="UTF-8" ?><   RootElement  >
            <Element1    AtTrIbUTe1 = " This  whitespace   should
appear as is."    >  This     text
        should all          be
    on one line!
</Element1 ><    Element2      name="VALUE"   name2="VAlue2"/><
Element3   attr
=
"Element1"   />

            <Element1    AtTrIbUTe1 = " This  whitespace   should
appear as is."    >  This     text
        should all          be
    on one line
    too!  Watch out for "quoted" content.
</Element1 ><    Element2      name="VALUE"   name2="VAlue2">  Nested
content< Element3   attr
=
"Element1"   />

anyone?
</   Element2>

        <operation name="process">
            <soap:operation soapAction="" style="document"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

</  RootElement   >
```

Program Output

Your program should print the XML from the input file in a more human readable format. Specifically, your program should:

1. Trim all whitespace from the beginning and end of the XML file.
2. Print the XML declaration as-is by itself on the first line.
3. Start each XML element on its own line. If an element is nested, indentation should be used to visually show how deep the indentation is. Use four periods per level to show the indentation inserted by your program. There is no limit to the level of nesting possible in an XML file.
4. Self-closing tags and elements with no nested elements in their content should be printed on a single line with no content following the end tag.
5. For elements with nested elements in their content, print the end tag at the same level of indentation as the start tag. In this case, both the start tag and the end tag should appear on their own line, with the content indented.
6. Leave no whitespace between the beginning of any XML tag and the XML element name.
7. Leave no whitespace between the greater than sign at the end of a start or end tag and the character preceding it.
8. Make sure there is one space between the forward slash at the end of a self-closing XML tag and the character preceding it.
9. Leave no space on either side of the equal sign of an attribute.
10. Make sure attributes are separated from XML element names and other attributes by a single space.
11. Change all attribute names to lowercase.
12. Remove whitespace between any element's start and end tags and text content. Also remove whitespace between XML tags.
13. Condense extra whitespace within an element's text content to a single space.

Problem 15: Pretty Print
30 points

Example Output:

```
<?xml version="1.0" encoding="UTF-8" ?>
<RootElement>
....<Element1 attribute1=" This  whitespace   should     appear as
is.">This text should all be on one line!</Element1>
....<Element2 name="VALUE" name2="VAlue2" />
....<Element3 attr="Element1" />
....<Element1 attribute1=" This  whitespace   should     appear as
is.">This text should all be on one line too! Watch out for "quoted"
content.</Element1>
....<Element2 name="VALUE" name2="VAlue2">
........Nested content
........<Element3 attr="Element1" />
........anyone?
....</Element2>
....<operation name="process">
........<soap:operation soapaction="" style="document" />
........<input>
............<soap:body use="literal" />
........</input>
........<output>
............<soap:body use="literal" />
........</output>
....</operation>
</RootElement>
```

Problem 16: Treasure Hunt
60 points

CODE QUEST 2014

Java program: `Prob16.java`
Input File: `Prob16.in.txt`
Output: Your output needs to be directed to stdout (i.e., using `System.out.println()`)

Introduction

Treasure hunting is a lost art form because of all this newfangled technology that everyone uses today. Did Captain Hook use Google Maps to find his way around the Neverland seas? No. He read a map. And so will you in this problem. Happy hunting! Oh, and bring a torch because it's dark in there.

Program Input

The file `Prob16.in.txt` will contain a single treasure map. Here is the key to reading the map and finding the treasure:

- H is the starting position of our treasure hunter (that's you). Don't get too lost!
- T is the treasure that you are trying to find. You win by stepping on this space.
- t is the symbol for a torch. You start out with 15 steps worth of torch light, and every torch you find gives you 15 more. The torches are unlit when you collect them, so assume you light them just as the previous torch is going out. So, picking up a torch on your first step would mean that you would have 29 more steps worth of light (15 to start + 15 for picking up a torch -1 for the first step). Each torch can only be collected and used once. You must step on a torch's space to collect it.
- x is an immovable and impassable barrier, like a cave wall. You can't step on an x.
- | and - denote the barrier of the cave. You cannot step on these spaces either. Don't worry about how you got in the cave or how you'll get out – finding the treasure will distract you from things like how to survive in a cave that apparently has no exit. ☺
- Stepping on a space more than once is permitted (and in some cases required – see the example problems). However, as previously mentioned stepping on a torch space twice will only give you extra light the first time.
- You can move up, down, left, and right only. Moving to a diagonal space takes two moves.

Problem 16: Treasure Hunt
60 points

CODE QUEST 2014

Example Input:

```
----------------------
|xxxxt   x  xxxxxxxxx|
|xxx     x xxx   Txxx|
|xx  xx     x    xxxx|
|xx   xx xx    xxxxxx|
|xxx  xxxxxxxxxxxxxx|
|xxx  xxx    t  xxxxx|
|xxx xxxx x xxx  xxxx|
|xxt        xxxx  xxx|
|x  xxxxxxxxxt   xxxx|
|xxxxxt    xxxx  xxxx|
|xxx  xx  xxx   xxxxx|
|xxxxx        xxxxxxx|
|H      xxxxxxxxxxxxx|
----------------------
```

Program Output

Your program should output a single integer: the least number of steps required to reach the treasure before (or just as) your light runs out. You can win the game by stepping on the treasure with your last step of light.

Example Output:

```
73
```

Additional Maps

While we were solving this problem, we found ourselves making maps to try out different situations. To spare you the difficulty of typing in a bunch of maps on your own, we have provided several of our test maps in the example input zip file. Below is a table containing the map names, the least number of steps, and the time it took our solution to run (just for reference, of course). The judging software does have time and memory constraints, so be careful. If you attempt this problem, we strongly suggest that you run these maps through your solution to make sure you can handle them. The judging map will strain your code more than any of these.

Map Name	Least # Steps	Execution Time (seconds)
SimpleTH.in.txt	14	0.0
SecondSimpleTH.in.txt	8	0.0
HardTH.in.txt	67	0.46
SuperHardTH.in.txt	56	0.78
LotsOfTorchesTH.in.txt	46	0.312
FourStepsTH.in.txt	60	0.0
UltraHardTH.in.txt	119	0.983