problem 1

X Liters of Ginger Soda

2 points

Introduction

A chef who owns his own restaurant has earned a reputation for creating fabulous new dishes by combining unusual ingredients. He has formulated a new recipe to release in conjunction with the opening of two new restaurants. The recipe uses ginger soda as one of the ingredients of a mushroom sauce, and the chef is very particular about the quality and flavor of the ginger soda. He found a supplier in Europe that produces a high-quality ginger soda he wants to purchase, but they only sell it in liters and he has specified the required quantity in gallons.

Write a program to convert liters to gallons. There are 3.785 liters in one gallon.

Sample Input

The input will be a single integer representing a number of liters.

```
144
```

Sample Output

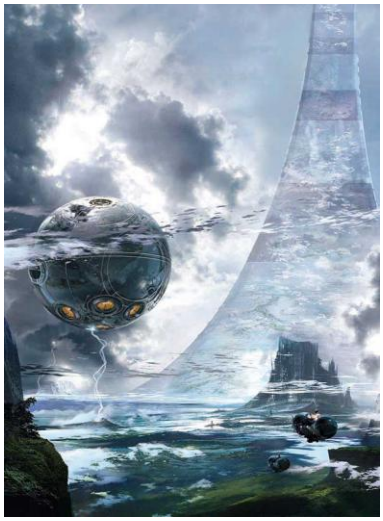
The program must print the equivalent number of gallons, rounded to the nearest integer.

```
38
```

problem 2
Ringworld
3 points

Introduction

In his science fiction novel Ringworld, Larry Niven described an enormous artificially constructed ring orbiting a star. The ring's radius was about 95 million miles and it was nearly 1 million miles wide. The side that faced inward was covered with land and seas and was surrounded by walls 1,000 miles high to retain the atmosphere. The supposed purpose of such a structure was to provide a habitable region with thousands or even millions of times more surface area than an Earth-sized planet.



Write a program to compute the inner surface area of a Ringworld and print the result relative to the surface area of the Earth. To do this you will need to know a few simple facts:

- the surface area of the Earth is 196.935 million square miles
- the surface area of a ring is 2·π·r·w, where r is the ring's radius and w is the ring's width
- the value of π is about 3.14159265

Sample Input

The input will consist of two real-number values. The first is the radius of the Ringworld, and the second is its width. Both values are given in miles. You might want to use a double data type to ensure your program doesn't lose any precision during calculations.

Example 1

```
95000000 997000
```

Example 2

```
92955887.6 131072
```

Sample Output

The program must print the inner surface area of the Ringworld relative to the surface area of the Earth. The value must be truncated to an integer value, accurate within +/- 1, followed by the word EARTHS. As you can see below, the Ringworld in the first example (which is the one described in the novel) would have a little over three million times the surface area of the Earth.

Example 1

```
3021869 EARTHS
```

Example 2

```
388726 EARTHS
```

problem 3

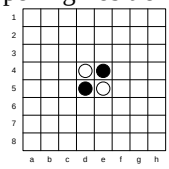
Reversi Moves

4 points

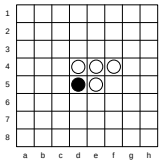
Introduction

Reversi is a game played between two players on an 8x8 grid. The opening position is shown below. White moves first. On White's turn, White must place a white piece on an empty square, such that there is at least one black piece between the new piece and another white piece, horizontally, vertically, or diagonally. After placement, all such black pieces (linearly between the new piece and the next white piece) are changed to white. The pieces must be adjacent in a single line, without empty squares. Black then plays similarly, converting one or more lines of white pieces. Play continues until either the board is filled or one player cannot make a legal move. An example of a game's first few moves is shown below.

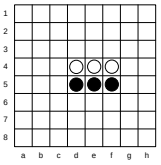
Opening Position



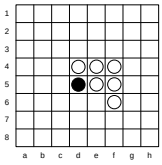
Move 1: f4



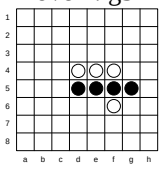
Move 2: f5



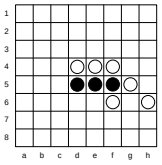
Move 3: f6



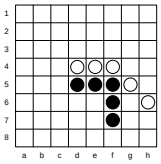
Move 4: g5



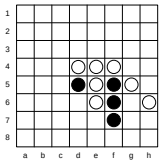
Move 5: h6



Move 6: f7



Move 7: e6



Your program should analyze a sequence of moves for a Reversi game. The opening position is shown above. You do not need to check the validity of each move.

Sample Input

The input will be a sequence of coordinates representing the placement of pieces for each turn (white places first.) "END" terminates the sequence.

```
f4
f5
f6
g5
h6
f7
e6
END
```

Sample Output

Your program should output the state of the board to the screen after all moves are completed. Use 'B' for Black, 'W' for White, and '.' (period) for empty squares. *Hint: printing the state of the board after each move may help you debug your program.*

```
........
........
........
...WWW..
...BWBW.
....WB.W
.....B..
........
```

problem 4

Return of Spell Binder

4 points

Introduction

After his last defeat at the hands of Letterman, Spell Binder was sentenced to a year of prison time for vandalism of public words. During his time away he only grew more determined to sow his own peculiar brand of havoc. Spell Binder has created an army of speelbots whose dastardly purpose is to replace letters in words being used in public places. Letterman discovered one of these speelbots at Kelly's Custard stand, where the bot changed the CUSTARD into MUSTARD. That doesn't taste very nice at all, precious. Fortunately for Kelly and all of her customers, Letterman was able to change the M back into a C. However, Letterman cannot fight an entire army of speelbots on his own, nor does he have any computer skills to write his own bot.

Write a program that can undo the havoc caused by Spell Binder's speelbots.

Sample Input

The input will consist of a single word followed by two groups of one or more letters, all separated by spaces.

Example 1

```
MUSTARD M C
```

Example 2

```
JUNK J TR
```

Example 3

```
MONSTER ON A
```

Sample Output

The program must correct the input word by replacing the letters of the first group with the letters of the second group and print the corrected word.

Example 1

```
CUSTARD
```

Example 2

```
TRUNK
```

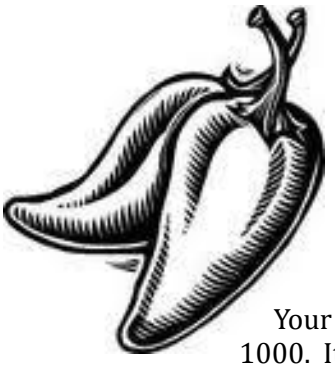
Example 3

```
MASTER
```

problem 5

Peter's Peppers

5 points



Introduction

Peter's Popular Prime Pepper Plant provides packs of peppers in packages of 6, 11, or 13 peppers. The price to prepare each package is the same, regardless of size.

Your program should take as input an integer less than 1000. It should find the cheapest combination of packages to ship that number of peppers.

Sample Input

The input will consist of a single integer, representing the count of peppers.

Example 1

```
42
```

Example 2

```
55
```

Example 3

```
27
```

Example 4

```
88
```

Sample Output

The program should display the cheapest combination of packages to ship the count of peppers. If the same minimum number of packages can be obtained in two ways, choose the one that uses more of the size-13 packages.

Example 1

```
42 peppers can be packed most economically in:
1 package of 13
1 package of 11
3 packages of 6
5 total packages.
```

Example 2

```
55 peppers can be packed most economically in:
5 packages of 11
5 total packages.
```

Example 3

```
27 peppers cannot be packed.
```

Example 4

```
88 peppers can be packed most economically in:
5 packages of 13
1 package of 11
2 packages of 6
8 total packages.
```

problem 6
Heronian Rectangles
5 points

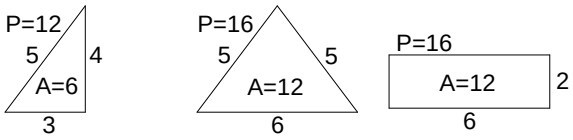
Introduction

Heron's Formula for the area of a triangle with sides of (a, b, c) is:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \qquad s = (a+b+c)/2$$

A Heronian Triangle is a triangle where all values above (A, a, b, c) are integers.

For CodeWars, define a Heronian Rectangle as a rectangle with integer sides with the same area and perimeter as a Heronian Triangle. For example, the smallest Heronian Triangle is (3, 4, 5), but there is no corresponding Heronian Rectangle. The Heronian Triangle (5, 5, 6) has a perimeter of 16 and an area of 12. The corresponding Heronian Rectangle has side-lengths of (2, 6).



Your program should find all pairs of Heronian Triangles and Rectangles, for a provided range of the longest side of the triangle, up to a maximum length of 500.

Sample Input

The input will be two integers representing the range for the longest side of the Heronian Triangle:

Example 1
```
6 13
```

Example 2
```
12 12
```

Example 3
```
490 500
```

Sample Output

Your program should output all triangle-rectangle pairs for the given range, one pair per line. For each pair, print the Triangle's side-lengths in increasing order, followed by the Rectangle's side-lengths in increasing order.

Example 1
```
(5, 5, 6) (2, 6)
(10, 10, 12) (4, 12)
```

Example 2
```
(10, 10, 12) (4, 12)
```

Example 3
```
(410, 410, 492) (164, 492)
(415, 415, 498) (166, 498)
```

problem 7
ASCII Skyline
6 points

Introduction

It's 1983 and you've just been hired by a videogame company (they're in "stealth mode", so you won't mention their name), and have been tasked with generating the backgrounds for their exciting new MS-DOS game. The game will use the state-of-the-art graphics, so you'll need to create some spiffy backgrounds to match the action on screen. The action takes place in the downtown area of a large Texas city - full of skyscrapers.

Because the game is dynamically building each "level" of the game, you can't just draw the backgrounds ahead of time, they must be generated on the fly. And to maintain the 3-D effects this hi-tech game requires, the buildings must be rendered in order from back to front.

Write a program that takes a list of building dimensions, and generate a skyline view of the city as output.

Sample Input

The program input consists of the number of buildings to render (maximum of 16), followed by a list of building dimensions. Each building dimension consists of: 1) the X-coordinate of the left edge of the building, 2) the width of the building, and 3) the height of the building.

```
4
4 4 8
10 10 4
13 5 6
2 4 4
```

Sample Output

The program will output the rendered skyline, using dash "-" for the roof of the building, a vertical bar "|" for each side, and hash "#" for the interior windows. The buildings are rendered back to front, such that buildings may be partially (or completely) hidden by later buildings. The output must also include a numbered X-axis wide enough to contain the image (continue printing to the next '0'). The maximum image size will be 60 columns by 20 rows, as we must leave room on the screen for the score and the command line!

```
   ----
   |##|
   |##|      -----
   |##|      |###|
 ----#|   ---|###|--
 |##|#|   |##|###|#|
 |##|#|   |##|###|#|
 |##|#|   |##|###|#|
12345678901234567890
```

problem 8

Letter Distribution

6 points

Introduction

The use of letters in the English language is not evenly distributed. For example, the letters E and T are used far more often than the letters X and J. In fact, the same principle holds true for any language in wide use. This same idea of inequitable distribution among a population can also been in in such disparate examples as the number of bids for items for sale on an auction web site, the distribution of wealth among people, and the scores of Code Wars participants. The study of distributions allows mathematicians to understand and work with dynamics and behaviors of large populations. It also helps computer system architects to design efficient systems for use by a large number of people.

We'll explore the idea of inequitable distributions by writing a program to display a histogram of letter occurrences sorted by popularity.

Sample Input

The input is a body of English text, up to 80 characters per line. The end of input is signaled by a single line with the string "###".

```
I have a dream that one day this nation will rise up and live out the true
meaning of its creed: "We hold these truths to be self-evident, that all men
are created equal." I have a dream that my four little children will one day
live in a nation where they will not be judged by the color of their skin but
by the content of their character.
###
```

Sample Output

The program must count the number of occurrences of each letter of the input and sort the letters by popularity, from most popular to least. Upper case and lower case letters are considered the same for counting purposes. Spaces and punctuation are to be ignored. Two or more letters with equal popularity must be sorted alphabetically. The program must print a horizontal histogram of the sorted letter counts as shown below so that one "*" is displayed for each occurrence of a letter.

```
E ***************************************
T ******************************
A ***********************
I ********************
H ******************
L ******************
N ******************
R ****************
O ***************
D ************
U ********
C *******
S *******
Y ******
B *****
F *****
M *****
V *****
W *****
G **
J *
K *
P *
Q *
X
Z
```

problem 9

Coderian Cipher

6 points

Introduction

Our ongoing battle with the Coderians is in full force. Every day, they continue to undermine our efforts by stealing our best lunchtime offerings (yes, it's a restaurant war.) Each morning their headquarters sends an encoded message to their stores giving them the instructions for one of our newest specials.

We've managed to receive a copy of today's message. Also, with the aid of a high-powered telescope and a fluttering window curtain, we were able to see the keyboard where the decoded message was being typed. Unfortunately, we only recorded a few successive letters before our view was blocked. We know the code is a simple alphabet-shift (for example, with a 3 letter shift, 'A' becomes 'D', 'B' becomes 'E', etc.), but we don't have time to try all possibilities. Hopefully, you will be able to decode the message so we can quickly promote our latest creation and expose their espionage before lunch.

Your program should use the series of decoded letters to completely decode the encoded sentence.

Sample Input

The input will consist of an alphabet-shifted encoded sentence of capital letters. The last two words will be the word KEY, followed by the captured series of decoded letters. The key letters follow each other consecutively, but their location could be anywhere in the sentence, from within a single word or across words (disregarding spaces.)

```
XYVOIC ERH FSCWIRFIVVC SR VCI AMXL E WMHI SJ XYRE WEPEH
KEY YONR
```

Sample Output

Your program should use the key letters to determine the size of the alphabet shift, then output the complete decoded message.

```
TURKEY AND BOYSENBERRY ON RYE WITH A SIDE OF TUNA SALAD
```

Here, the key "YONR" is seen spanning the third through fifth words of the Sample Output. This code had a shift of four letters: i.e. every "A" became "E", and every "Y" became "C". Decoding required shifting back four letters.

problem 10
Card Counting
7 points

Introduction

Casinos go through many decks of cards and are probably interested in periodically verifying that all expected 52 cards are there and no extras have managed to slip into the deck. You have decided to help them out with this problem, for a nominal fee.

Write a program that takes in a list of cards and then determines what, if anything is wrong with the deck. Missing cards (if any) should be listed first, followed by extra cards (if any) and the number of extras of that type. Output should be sorted by rank (2-A) and then suit (spades, hearts, diamonds, clubs) order.

Sample Input

The input is a set of lines of cards in the deck, not necessarily in order. Suits are abbreviated as follows: H=hearts, D=diamonds, C=clubs, S=spades. Thus, for example, "7S" is the seven of spades. The first line of input specifies the number of lines of card input to follow.

```
13
5D KC 3C 3D
10C 7D 4C AH
10S JH 6C 10D
7S 2H JS QH
2C 7C KD 8C
QD QS KH 2S
3H 4S 5H JD
3S 4H 5S 6H
8H 6S 7H QC
9C 8S 9H 6D
2D 8D AS 5C
JC AH 4D KS
AH 9S 10H 9D
```

Sample Output

```
Missing cards:
AD AC

Extra cards:
AH (2)
```

problem 11

Hardware Troubles

7 points

Introduction

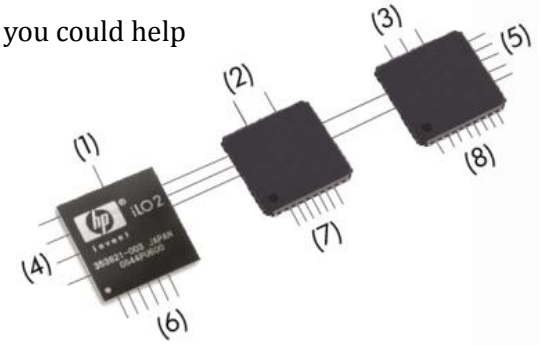
All the chips fell out of my motherboard…again. I don't want to talk about it. I know how many contacts there are on the edges, but I don't remember where each chip came from.

If you sum the number of contacts going into a given socket, you'd get the number of contacts on the chip that's supposed to go there. All contacts must be used.

But it gets really confusing really fast. Maybe you could help me out?

Here's an example board with three chips missing. Each 'x' represents a missing chip:

```
0 1 2 3 0
4 x x x 5
0 6 7 8 0
```



If we examine the board above:

- The leftmost chip x must have at least 11 contacts, since the sum of the surrounding numbers (1, 4, 6) is 11.
- The middle chip x must have at least 9 contacts, since the sum of the surrounding numbers (2, 7) is 9.
- The rightmost chip x must have at least 16 contacts, since the sum of the surrounding numbers (3, 5, 8) is 16.
- If the leftmost x were 12, then the middle x must be at least 10 because it would need to utilize one contact from the leftmost x.

Sample Input 1

The program input consists of 1) the width of the board in width/height dimensions, 2) the representation of the motherboard with empty sockets denoted with 'x', and 3) the list of loose chips, represented by their contact counts. 0's mark the corners of the chips and can be ignored.

```
4 4
0 8 7 0
1 x x 6
2 x x 5
0 3 5 0
9
14
6
12
```

Sample Output 1

The program should output the motherboard layout, with the missing chips placed in their appropriate sockets:

```
0 8 7 0
1 9 14 6
2 6 12 5
0 3 5 0
```

Sample Input 2

```
5 5
0  1   2   3   0
4  x   x   x   5
6  x   7   x   8
9  x   x   x  10
0 11  12  13   0
5
9
8
13
15
20
19
23
```

Sample Output 2

```
0   1   2   3   0
4   5   9   8   5
6  13   7  15   8
9  20  19  23  10
0  11  12  13   0
```

problem 12
Dueling Princes
8 points

Introduction

A number of aggressive princes decided that only one of them should be king. They agreed they would duel one another until only one remained. On the day of the proposed bloodbath, those who expected to win (that is, all of them) each realized his kingdom would be rather pitiful if he were the only inhabitant.

Instead, they chose to act out their duels in a board game. On a square board 1000 squares on a side, they each in turn placed their token on one empty square. All princes then followed the same logic for their moves: in the same order, each prince moved his token one square (in any direction, like a chess king) to minimize the distance to his nearest competitor. If he moved onto a competitor's square, the competitor was vanquished; the attacking prince kept his token on the square, and the defender removed his token and proceeded to sulk for the rest of the game. The princes continued to take turns in this way until a single king was determined.

Your program should read the princes' starting locations in the 1000x1000 grid and determine each duel and its location in the grid. The princes take their turns in the order provided. If a prince has more than one closest competitor, he will choose the one with the lower row number first, then the one with the lower column number. Distance is measured as a regular Euclidean distance between the center of squares:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Hint: Don't store the distance as an integer – you may truncate important data.

Sample Input

The first line of input is the number of princes. Each successive line includes a prince's name, row, and column location when the game begins.

Example 1

```
6
Thomas 999 999
Charles 998 998
Edward 1000 1000
Albert 998 1000
Marty  1000 998
Bruce 1 1
```

Example 2

```
5
Abe 1 1
Bob 1 2
Carl 1 3
Doug 1 4
Bruce 3 4
```

Sample Output

Your program should print the location, winner and loser of each duel, in the order they occur.

Example 1

```
(998, 998) Thomas defeats Charles
(999, 1000) Albert defeats Edward
(999, 998) Thomas defeats Marty
(999, 999) Thomas defeats Albert
(501, 501) Thomas defeats Bruce
```

Example 2

```
(1, 2) Abe defeats Bob
(1, 2) Carl defeats Abe
(1, 3) Carl defeats Doug
(1, 3) Bruce defeats Carl
```

problem 13
House Numbers
8 points

Introduction

The Ufinnish construction company completed the last house on the street yesterday, and you are due to start selling them at noon. Unfortunately, they left without installing house numbers. All they left was a box full of single-digit numbers and a screwdriver. You know you have exactly the digits you need, but you don't know what the actual numbers should be.

The houses are evenly arranged on both sides of a north-south street. The community planning commission creates some very odd house numbering schemes, but they do follow a few guidelines:

- Houses on the west side of a north-south street have odd numbers that increase from north to south.
- Houses on the east side have even numbers exactly one number greater than the odd-numbered house across the street.
- The difference between any two neighboring houses' numbers is always the same.
- Each house number uses from one to four digits (with no leading zeros.)
- If, somehow, all the house number digits on the street were discovered in a box, then of all possible sets of house numbers which could be created from that box for the street, the street starts with the smallest possible house number and continues with the smallest possible house numbers.

Your program should read the number of houses on the street (up to 40) and the count of each digit 0-9 in the box and determine the house numbers for each house.

Sample Input

The input consists of eleven integers. The first is the number of houses. The next ten are the count of each digit found in the box (count of 0's, count of 1's, etc.)

Example 1
```
10 1 10 1 1 1 1 1 1 1
```

Example 2
```
8 0 16 6 0 2 0 0 2 0 0
```

Example 3
```
18 3 5 4 3 3 4 4 4 7 12
```

Example 4
```
8 1 1 3 4 6 7 4 2 3 1
```

Sample Output

Your program should print the difference between neighboring houses, then all the house numbers in numerical order, up to ten numbers on each line.

Example 1
```
Common Difference: 2
9 10 11 12 13 14 15 16 17 18
```

Example 2
```
Common Difference: 700
11 12 711 712 1411 1412 2111 2112
```

Example 3
```
Common Difference: 98
1 2 99 100 197 198 295 296 393 394
491 492 589 590 687 688 785 786
```

Example 4
```
Common Difference: 1112
2345 2346 3457 3458 4569 4570 5681 5682
```

problem 14

Programming with a Lithp

9 points

Introduction

LISP is the second-oldest high-level programming language that is still in use (the oldest is Fortran). While not as popular as Java or C++, LISP holds a tenacious niche in such diverse applications as databases, graphics, and artificial intelligence. The language's name is a contraction of LISt Processing, which reflects its syntax. In LISP both code and data are written as lists contained in parentheses. So by contrast, a function call to compute cosine in Java is written cos(x) but in LISP the same call is written (cos x). There are no comma delimiters for list elements, so a function call to subtract seven minus two might be written like this: (sub 7 2). List expressions may of course be embedded, so the following two expressions would be equivalent:

LISP expression	algebraic expression
`(sub n (mul 2 (cos (add (mul 3 x) 1))))`	$n - 2\cos(3x + 1)$

As you can see, even a short program can contain a lot of nested parentheses. LISP programmers face a real challenge trying to keep track of parentheses, especially as they change nested code by adding or removing expressions. That's where you come in.

Write a program that can read a LISP expression and determine if all the function calls have the correct number of arguments.

Sample Input

The first input section lists the number of functions to follow, then each function name and its minimum and maximum number of arguments. An asterisk indicates no maximum. The second section consists of a LISP expression. You may assume there are no string constants, that all numbers will be non-negative integers, that the input will have correct pairing of parentheses, and that all tokens (function names, numbers, and variables) will be separated by white space (one or more spaces or line feeds).

```
3 SET 2 2 ADD 2 * MUL 2 *
( SET Y ( ADD ( MUL A X X ) ( MUL B X ) C ) )
```

Sample Output

If the LISP expression is valid the program should print "VALID EXPRESSION". If a function call has too many arguments the program should print "FUNCTION foo: TOO MANY ARGUMENTS". If a function call does not have enough arguments it should print "FUNCTION foo: NOT ENOUGH ARGUMENTS". There will be a maximum of one error in the input.

```
VALID EXPRESSION
```

More sample input/output...

```
6 SET 2 2 SUB 2 2 MUL 2 * COS 1 1 ADD 2 * DIV 2 2
( SET T0 ( SUB X1 ( MUL 2 ( COS ( ADD ( MUL 3 THETA ) 1 ) ) ) ( DIV X1 3 ) ) )

FUNCTION SUB: TOO MANY ARGUMENTS
```

problem 15

Queueing Theory

9 points

Introduction

Queueing theory is the mathematical study of waiting in lines. Doesn't that sound fun? In computing, queues are used to track requests that have been issued but not started. This is seen in applications like web services or online games in which a large number of people request data from a server. The order of request processing we will consider is called First-In First-Out or FIFO. With a FIFO the processing works like lines at a grocery store, where each line (queue) is filled with customers (inbound requests) and is serviced by a cashier (a time-constrained processor). Customers arrive at the back of the line and the cashier services the customer at the front of the line. In other words, FIFO.

Simulating a time-constrained process in a contest setting is awkward. Instead, we'll use a process that provides a simple way to verify that the FIFOs are working correctly. Your program will simulate system storage using a data string that has been initialized with spaces. Queued requests will consist of a word and an offset. The program must write the requested word into the data string at the designated offset. The first character in the data string is at offset zero. All requests write into the same data string buffer. As an example of a request, for a data string of length 44 the request "35 KNOWN" would result in the following data string:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
																																				K	N	O	W	N				

Each request is added to a request queue and processed later. Some requests may overwrite portions of words previously written in the data string, so sequence is important. Each queue is a FIFO, but just like at the grocery store multiple queues can mean that requests (customers) are not always processed in the order they arrived.

Sample Input

The first line contains the length of the data string and the number of requests. The following lines are the actual requests. Each request has three parts: a queue number, an offset, and a word. There may be up to nine request queues (amazingly, numbered 1 through 9). The last line of input designates the order in which the requests are actually processed from the queues.

```
44 13
Q1 35 KNOWN
Q1 20 IMPORT
Q3 24 GRANT
Q1 4 IN
Q1 15 MADE
Q1 32 AN
Q2 39 LEDGE
Q2 5 NOTION
Q2 6 A
Q2 16 OR
Q3 0 IMAGE
Q3 12 IS
Q3 30 THIS
Q1 Q3 Q3 Q3 Q2 Q1 Q2 Q1 Q3 Q1 Q2 Q2 Q1
```

Sample Output

The program must process each request in the designated order and print the result string.

```
IMAGINATION IS MORE IMPORTANT THAN KNOWLEDGE
```

problem 16

Escape The Vaults

12 points

Introduction

The intrepid explorer Codington Warson has finally made it to the inner vaults of his latest dig. Unfortunately, he discovered each room has an intricate trap to unravel. There is a pressure plate which opens the exit door to the room, but the plate is too far away from the door. There are large artifacts in each room which can be slid to cover the plate, but they can only be slid in certain directions, requiring him to make several moves to finally cover the plate.

Codington has brought his trusty HP Envy Sceptre laptop to provide him the most economical set of moves needed to escape the room and is just waiting for your program to help him. Luckily, Codington knows that this civilization was a rather lethargic group, so any solution will involve moving objects no more than 30 times.

Your program should analyze a diagram of the room and find the solution that uses the given number of moves.

Sample Input

The input provides a diagram of the room. The first line will hold three integers representing:

- The number of rows in the grid.
- The number of columns in the grid.
- The number of moves required to cover the plate.

The remaining lines will each provide a description of one row of the grid. A digit (1-9) marks an artifact which can only be moved left or right. A letter (a-j) marks an artifact which can only move up or down. The capital letter X marks the pressure plate. A period (.) marks an empty square. If two squares are labeled by the same number or letter, they indicate a large artifact which covers those squares and must move as one unit.

```
5 5 14
d3333
d.c.3
2acXb
1a..b
11..b
```

Sample Output

Your program should output on one line each individual move “xy”, where x is the artifact and y is the direction U/D/L/R, separated by spaces. It should then print an empty line and a diagram of the final configuration of the room.

```
aU 1R 1R aD aD 2R dD 3L bU 1R cD cD 2R 2R

3333.
d..3b
d..2b
.ac1b
.ac11
```

problem 17

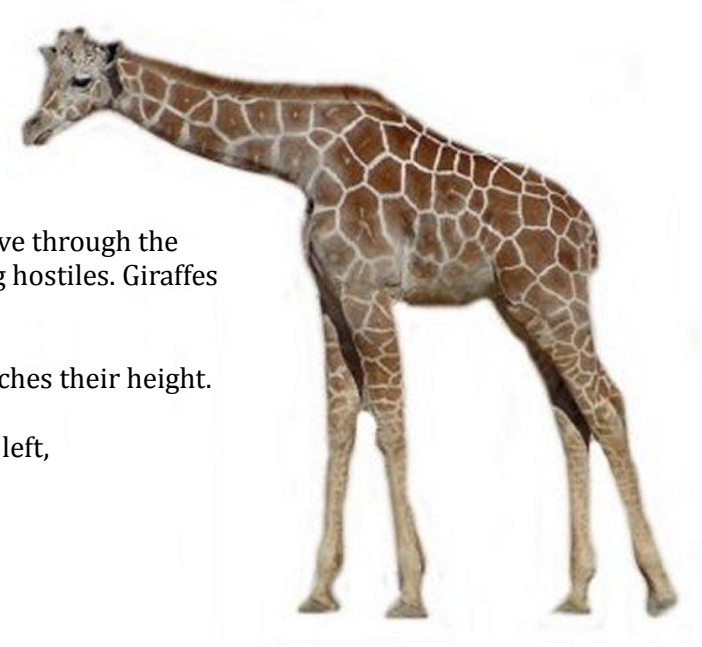
Giraffe Positioning System

13 points

Introduction

The military wants to use highly trained giraffes in their latest campaign. However, giraffes are extremely poor navigators. Your task is to create a Giraffe Positioning System with turn by turn directions.

You will be given a number of giraffes that you need to plan routes for, followed by a name and height (in feet) for each giraffe. Then, you will be told the dimensions of the field in sectors (height first, then width). The field will be denoted by numbers representing the height of the lowest branch (in feet) in that sector. Your giraffes will remove any branches up to their height, anything more would be too slow and the mission would surely fail. Each foot of delicious foliage consumed will slow your team as though they moved through one sector. The insertion point is denoted by an S and the mission objective by an F, these can occur anywhere in the sector.



Your goal is to select the paths so that the cumulative movements of the giraffes are minimized. In other words, it's possible that, for example, the first giraffe may take a less than optimal path in order to optimize the total travel time of the entire group. Only one Giraffe can move through the combat zone at a time. Anything else would risk alerting hostiles. Giraffes will be released in the order provided by the roster.

Giraffes will move into a sector and then eat until it matches their height.

U = move up, D = move down, R = move right, L = move left, and E = eat (once per foot of foliage).

Sample Input

```
3
Irving 10
George 12
Geoffrey 15
4 4
S   5 11 11
11  6 11 11
11 11 11 11
11 14 11  F
```

Sample Output

On each line, print the giraffe's name, the number of moves it made, and all of its moves separated by spaces. Then print the total number of moves for the team.

```
Irving, 6: D D D R R R
George, 10: D E D E D E R R E R
Geoffrey, 19:  D E E E D E E E D E E E R E R E E E R
Total moves: 35
```

problem 18

A-Maze-ing Portal

14 points

Introduction

Aperture Science Corp. is accepting volunteers to test their latest product, the Portal Gun. The volunteers will be tasked with traversing a traditional maze, with the scientists measuring their efficiency and use of the Portal Gun to create shortcuts.

Your job is to create a program to judge the volunteers' performance by finding the shortest path through the maze, using known pairs of portal entrances and exits (which allow the volunteer to move from one point to the other). Each volunteer is told that if they discover the shortest path, Aperture Science will hold a party in their honor, and there will be cake*.

Write a program which reads the description of a rectangular maze of unit squares and displays the shortest path through the maze. A path's length is simply the number of squares entered between Start and Finish. Portals can be used in either direction, and if a volunteer moves to a portal entrance, they **must** use the Portal Gun; their next step will be onto the associated exit of that portal. The portal then disappears, and both squares simply become empty squares of the maze (a portal can only be used once.) The input includes:

- 2 integers representing the number of rows (max: 25) and columns (max: 75) in the maze grid.
- Up to 25 lines, each describing a row in the maze grid. Each line is no more than 75 characters long.
  Within each line:
  - A hash "#" marks a square that cannot be visited.
  - A period "." marks an empty square that can be visited.
  - The letter "S" marks the Start square.
  - The letter "F" marks the Finish square.
  - Duplicate letters "A", "B", "C", or "D" represent the entrance and exit pairs for a portal.

Your program should print:

- The length of the shortest path, in squares, and the order any portals are used.
- A diagram of the shortest path. Reprint the input maze using:
  - spaces for empty squares
  - a period "." for each square visited along the path

**The cake is a lie.*

Sample Input

```
10 21
######################
#..B#...#.......#A..#
#.#.#.#...#####.###.#
#.#.#.#####F#.......#
#A#...#...#..B#.#.#.#
#.#####.#.#####.#.#.#
#.#.....#.......#...#
#.#.################.#
#S...................#
######################
```

Sample Output

```
Distance: 37 squares
Portals Used: AB
######################
#..B#    #        #A..#
#.# # #    ##### ###.#
#.# # #####F#       .#
#A#    #    #..B# # #.#
# ##### # ##### # #.#
# #      #        #  .#
# # ################.#
#S...................#
######################
```

problem 19
QweRty
15 points

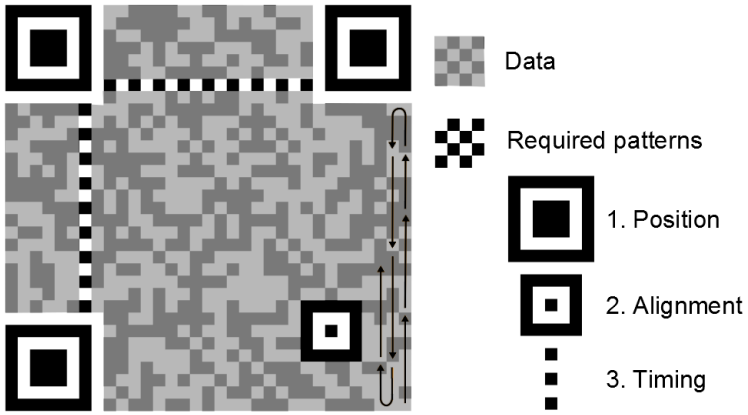
Introduction

QR codes are a popular way to encode data for easy reading by computerized cameras.

True QR codes contain error correction, format specifiers, version information, and a size variable layout that makes them harder to decode and process. So we've decided to create our own QR codes called QweRty codes that are simpler to decode and contain more data.

QweRty codes contain the same position markers as QR codes, but the intervening data is encoded differently. The new code is useful, but not compatible with QR codes, and so we need a new program to decode the information.

Each character of the text within the code is composed of 8 bits in standard ASCII encoding. A filled-in block denotes a 1 bit, a blank block denotes a 0 bit. Blocks are placed so that the most significant bit is first, and the entire message is read in serpentine fashion (up one column, then down the next, etc.), starting at the lower right. For example, in the Sample Input, 'W' (01010111b) can be read up the right column, followed by 'e' (01100101b).



The structure of the QweRty code is as described in the graphic. Your program should ignore the required patterns.

Your program will be provided a QweRty code as input, and must decode the contained message.
Each filled-in block is denoted by ## (two characters), and each empty block with 2 spaces.

Sample Input

```
##############    ####  ######        ######  ##    ##############
##          ##  ##  ##  ##          ##  ##      ##  ##          ##
##  ######  ##    ##    ##    ##  ##########  ##    ##  ######  ##
##  ######  ##    ######  ####    ##  ##  ########  ##  ######  ##
##  ######  ##  ##  ##  ##  ##  ##  ##  ####    ##  ##  ######  ##
##          ##  ##  ##  ##      ##  ##  ######  ##          ##
##############  ##  ##  ##  ##  ##  ##  ##  ##  ##############
                ##                  ##  ##  ##
            ##  ##  ##  ##    ########  ####  ##        ##  ####
                ####        ########    ####    ####  ##  ######
          ########      ##      ##  ##          ####    ####
              ######  ##  ##  ######  ##  ##          ##  ##    ##
            ##      ############  ##  ##  ##  ##          ##    ##
                ####  ####  ##  ##      ##      ######  ####
              ######  ##  ##  ##########  ##      ##    ####    ####
            ##          ##      ##  ##  ##  ##      ##  ##  ####  ##
              ########  ##    ####  ####              ##      ####
                  ##    ######  ##  ######  ##      ######    ######
              ####        ##      ##  ##              ##  ####
                ##  ##  ##  ##  ##  ##  ##  ######              ####
              ####  ##  ##  ##  ##########      ##  ####        ##
                    ####      ##  ##    ################  ##    ##
              ##  ##    ########  ######      ##    ####  ####  ####
                ##  ##      ##  ##  ##          ##      ##        ##
              ####          ####        ######  ############  ####
                        ##    ##    ##  ##      ####      ##    ####
##############          ##              ##  ##  ##  ##      ##
##          ##    ######  ##  ##  ##    ####  ####      ##########
##  ######  ##    ##  ##  ######  ##  ######################  ##
##  ######  ##  ##########      ##  ##  ######          ##  ##  ##
##  ######  ##    ####      ##          ##  ####  ##########  ##
##          ##        ##  ##  ##  ##      ##    ####  ######    ##
##############  ##          ##  ##  ##      ##        ####  ####
```

Sample Output

```
Welcome to CodeWars and Fun with QweRty Codes! Welcome to CodeWars and Fun with QweRty Codes!
```

2012 HP Code Wars

Event Evaluation

We need your input to evaluate our efforts, and, therefore, ask that you complete the following questionnaire.

Please rate the following	**1 Great**	**2 Pretty Good**	**3 Okay**	**4 Fair**	**5 Poor**
Overall competition					
Parking					
Registration process					
Competition Instructions					
Competition Room (set up, comfort)					
Time allotted					
Give-aways-Event Bags					
Door Prizes					
Lunch					
Judging					
Divisional problems					

What was your favorite part of this event?

_____

What would you change about the event?

_____

Other comments:

_____

_____

_____

I am a Student ☐ Teacher ☐ representing _____ High School

Email: _____

2012 HP Code Wars

Event Evaluation

We need your input to evaluate our efforts, and, therefore, ask that you complete the following questionnaire.

Please rate the following	**1 Great**	**2 Pretty Good**	**3 Okay**	**4 Fair**	**5 Poor**
Overall competition					
Parking					
Registration process					
Competition Instructions					
Competition Room (set up, comfort)					
Time allotted					
Give-aways-Event Bags					
Door Prizes					
Lunch					
Judging					
Divisional problems					

What was your favorite part of this event?

_____

What would you change about the event?

_____

Other comments:

_____

_____

_____

I am a Student ☐ Teacher ☐ representing _____ High School

Email: _____

2012 HP Code Wars

Event Evaluation

We need your input to evaluate our efforts, and, therefore, ask that you complete the following questionnaire.

Please rate the following	**1 Great**	**2 Pretty Good**	**3 Okay**	**4 Fair**	**5 Poor**
Overall competition					
Parking					
Registration process					
Competition Instructions					
Competition Room (set up, comfort)					
Time allotted					
Give-aways-Event Bags					
Door Prizes					
Lunch					
Judging					
Divisional problems					

What was your favorite part of this event?

_____

What would you change about the event?

_____

Other comments:

_____

_____

_____

I am a Student ☐ Teacher ☐ representing _____ High School

Email: _____