PYTHON PROJECT 1

Indexing: I have challenged myself to complete the following exercises without indexing.

Exercise 1 (r21.py):

Write a function r2l(text) to print a string on successive lines from right to left. So, for example

```
>>> r2l ( " backwards " )
        b
       a
      c
     k
    w
   a
  r
 d
s
```

Use your function to print the word slantwise diagonally from right to left.

Exercise 2 (ring.py):

The aim of this question is to draw a ring of stars like the European Union flag using the turtle graphics in exturtle.
First write a function star(turtle, x, y, points, R, r) which uses turtle turtle to draw a star centred at (x, y). The points argument should specify the number of points the star has. The R and r arguments specify the outer and inner radii of the star respectively. The outer radius R is the distance from the centre of the star to the points and the inner radius r is the distance from the centre to the corners in between the points. The Appendix gives some more information on the geometry of stars and hints for drawing one.
Use your function to draw a row of stars with 5, 6, 7 and 8 points.
Write a second function ring(turtle, cx, cy, Nstars, radius, points, R, r) which draws a ring of Nstars stars, so that each star is a distance radius from the centre of the ring at (cx, cy). The points argument should specify the number of points for each star and the arguments R and r specify the outer and inner radii of each star. Your ring function should use your star function to draw each star.
Use your ring function to draw an EU flag with 12 5-pointed stars.

Exercise 3 (prize.py):

You are given credit C which you can spend on two prizes at WinTwoPrizes.com. However, you may only spend the credit on exactly two items and the cost of these items must exactly equal C.
You are given a list of the costs of the available items. This list is guaranteed to be in ascending order. You should write a program, prizes.py, that prints out the costs of the pairs of prizes you could choose. If there is more than one combination of prizes whose costs add up to your credit, you should print all the possible combinations that you find. If no two costs add up to the credit you have, then your program should print Hard luck.
The data you are given will be in the following format:

- The first line gives the credit you have. This will always be a positive integer.
- The remaining lines give the costs of the prizes. Each of these is also a positive integer. There will always be at least two costs.

Make sure that your program outputs only the lines containing the positions in the list or a single line containing Hard luck. Prizes that may be won may be listed in any order.

Exercise 4 (validate.py):

Credit card numbers are constructed in such a way that it can be verified from the numbers alone whether the number is a valid credit card number. This can be helpful when transmitting the number, for example: if the received number is not valid, there must have been an error in transmission.
The algorithm for verifying a number is as follows:

1. Starting with the penultimate digit, and working towards the first digit, double alternate digits.
2. Sum the doubled digits, treating 13 as 1 + 3, etc, and add the result to the sum of the undoubled digits
3. If the sum is divisible by 10 the number is a valid credit card number.

A credit card number is received as the string "48X926742" where the X indicates that there was an error receiving the third digit. By writing a loop that tries all

the possible numbers, use your
is valid function to discover the correct number.

Exercise 5 (trains.py):

This problem is taken from the UK and Ireland Programming Contest held last year.
The Exeter Trains is losing money at an alarming rate because most of their trains are empty. However, on some lines the passengers are complaining that they cannot fit in the carriages and have to wait for the next train!
Exeter Trains want to fix this situation. They asked their station masters to write down, for a given train, how many people left the train at their station, how many went in, and how many had to wait. Then they hired your company of highly paid consultants to assign properly sized trains to their routes.

You just received the measurements for a train, but before feeding them to your optimisation algorithm you remembered that they were collected on a snowy day, so any sensible station master would have preferred to stay inside their cabin and make up the numbers instead of going outside and counting.
Verify your hunch by checking whether the input is inconsistent, i.e., at every time the number of people in the train did not exceed the capacity nor was below 0 and no passenger waited in vain.
The train should start and finish the journey empty, in particular passengers should not wait for the train at the last station.

Information about a train is contained in a file. The first line contains two integers C and n
(2 <= n <= 100), the total capacity and the number of stations the train stops in. The next n lines
contain three integers each, the number of people that left the train, entered the train, and had to
stay at a station. Lines are given in the same order as the train visits each station.
Write a program to read the file and output a single line containing one word: possible if the
measurements are consistent, impossible otherwise.