Question 1: (15 points)

Write a function named `pyramid(s)` that takes a string as its input

and PRINTs out a pyramid on the screen based on this string. For example,

when called as `pyramid("BURKAY")` it will print:

```
B
UU
RRR
KKKK
AAAAA
YYYYYY
"""
def pyramid(s):
  # remove the following line to solve this question
  return
"""
```

Question 2: (15 points)

Write a function named `pairs(a,b,c,d)` that takes four

integers as its input and checks whether the sum of any

two of these integers is equal to the sum of the other two.

If so, the function RETURNs this sum. Otherwise, it RETURNs -1.

For example, `pairs(1,2,3,4)` should return 5,

because 1 + 4 = 2 + 3 = 5.

Another example, `pairs(11,22,13,20)` should return 33,

because 11 + 22 = 13 + 20 = 33.

Another example, `pairs(1,2,3,5)` should return -1,

because no such two pairs exist among these numbers.

"""

```
def pairs(a,b,c,d):
  # remove the following line to solve this question
  return
"""
```

Question 3: (25 points)

Write a function called `filler(A, l)`

that takes an integer `A`, and a list of integers `l`

and if some of the numbers in `l` sum up to `A` then

it returns a list of these numbers. Otherwise, it returns

`None`. For example,

filler(5, [1,6,3,4]) returns [1,4].

filler(5, [1,6,3,7]) returns None.

filler(0, [-1,-1,2,-3]) returns [-1,-1,2].

filler(10, [2,3,2,3,1]) returns [2,3,2,3].

If there are multiple answers, then return any of them.

You can use each number in the list once for each occurence.

```
"""
def filler(A, l):
  # remove the following line to solve this question
  return
"""
```

Question 4: (20 Points)

Write a function named `vectorSum(l1, l2)` that takes two lists of numbers and adds the numbers in l2 on l1, one by one. If l2 is shorter than l1, then l2 recycles from the beginning once it is over. If l2 is longer than l1, then only the first part of l2 that matches the length of l1 is considered. Your function must return the summed up resulting list. For example, vectorSum([1,2,3,4,5], [1,2,3]) returns [2,4,6,5,7] because:

```
1,2,3,4,5
+ + + + +
1,2,3,1,2    -> Here, l2 recycles from the beginning
= = = = =
2,4,6,5,7
```

vectorSum([1,2,3], [1,2,3,4,5]) returns [2,4,6] because:

1,2,3

+ + +

1,2,3,4,5   -> Here, only the first 3 items of l2 are used

= = =

2,4,6

vectorSum([1,2,3,4,5], [1,2,1,2,1]) returns [2,4,4,6,6] because:

1,2,3,4,5

+ + + + +

1,2,1,2,1   -> Here, length of l2 matches the length of l1

= = = = =

2,4,4,6,6

You can assume that l1 and l2 are not empty.

"""

```
def vectorSum(l1, l2):
  # remove the following line to solve this question
  return
```

"""

Question 5: (25 points)

Write a function named `freq(l)` that takes a list `l` of numbers

and returns a tuple of the most frequent number and its frequency.

For example, `freq([1,2,3,4,5,2,3,4,5,3,4,5,4,5,4]) returns (4,5)

because number 4 appears 5 times in this list. If multiple numbers

appear as most frequent, then the first element of the returned

tuple is a list of all those numbers. For example,

freq([1,2,3,1,2,1,2,3,1,2]) will return ([1,2], 4) because both

```
1 and 2 appear 4 times in the list.

"""

def freq(l):
  # remove the following line to solve this question
  return
```