Week 5 Homework:
#1 - A) In order to implement a backtracking algorithm, you need to start with the target sum and work backwards. You can treat 1-9 as an array of length 9, and from within that
recursively call a function to continue to add the next digit in the array. Calculate the
difference from the target at each step, and if it is below 0 you know to backtrack since you have gone above the target. If it is 0, that means you are at the target and you only need to check if the current combination is the correct length needed for the solution. After you run through this for loop once making recursive calls you will iterate and start again, but beginning with the next index of the array, making
this very time inneficient.

#1 - B) Psudocode:

```
function combination sums(length, target):
    result = [empty array]
    key = [1,2,3,...,9]
    helper_function(key,start,result,target,[],length)
    return result

helper_function(arr,start,result,target_sum,combo,length):
    if target_sum == 0
        check right length, if so append to result
        return   #return either way, since if not right length not an answer
    elif remainder less than 0:
        return
    for i in range(start,len(arr)):
        append i to combo
        helper_function(arr,i+1,result,target_sum-arr[i],combo, length)
        combo.pop()
```

#2 - A) To start off you need to realize that you under no circumstances can feed more dogs
than you have treats. Because of this, you need to be able to track the treats you are trying to
feed to dogs. You first need to sort your treats array, since the dogs is already sorted by hunger level. Track which dog you are on, starting at index 0, and then you need a for loop to iterate over the treats, comparing them to the first and least
hungry dog. Since you are starting with the smallest treats, if/when you find one that is
of equal size of larget you know that its the smallest possible one for the first dog. Then
you increment the dog counter, and check the next treat, since you know all of the previous ones wont work for the next dog. Return the counter when done with the treats
for how many dogs you fed.

#2 - C) The sort function built into python is O(nlogn) and the for loop is just

O(n)
so the time complexity for this would be O(nlogn) as that is the most signifigant.

DEBRIEFING:
1) 10-12
2) Moderate, I am having difficulty with the same aspects I have had from earlier in the course.
The new material I am able to grasp, but calculating and knowing time complexities is hard.
3) 75%