10-9 Lecture:

QOTD13: his answer

```
def countVC(S):
    result = i = 0
    while i < len(S):
        while i < len(S) and S[i].lower() not in 'aeiou':
            i = i +1
        if i < len(S):
            result = result + 1
        while i < len(S) and S[i].lower() in 'aeiou':
            i = i+1
    return(result)
```

Lab5:

1. np(S)

```
def np(S):
    return (S[1:] == S[1:][::-1]) or (S[:-1] == S[:-1][::-1])
    #return(len(S) >= 1 and S[1:]==S[1:][::-1] or S[:-1]==S[:-1][::-1]
```

2. iqp(S):

```
def iqp(S):
    for i in range(len(S)//2):
        if S[i] != S[len(S) - i -1]:
            return(np(S[i:len(S)-i]))
    return(True)
#wow perfect xd
```

3. rqp(S):

```
def rqp(S):
    if S[0] != S[-1] or len(S) < 2:
        return np(S)
    return rqp(S[1:-1])
```

```
If len(S) < 2 or S[0] != S[-1]:
return(np(S))
```

4. cntDiff(S1, S2):

Review: Algorithmic Analysis:

- Question: how many times can I (integer) divide N by 2 before I get to 1?
- Answer: log2(N) or simply log(N)
- Note: recursive nature of the implementation has nothing to do with the analysis
- Comprehensions are bounded iteration used to construct new mutable structureL lists dictionaries etc
- For is used for bounded iteration when not constructing or collecting new structure or where early termination may be necessary
- While is just like for unbounded iteration rather then ranging over a sequence of known length
- Recursion is the most flexible/ general of them all: not every recursive function maps onto a nice iterative equivalent.
- To see why this is so, imagine a recursive implementation of sum that also handles embedded lists of elements within the argument

The Word Ladders Game:

- The challenge is to take 2 words o the same length and produce a chain of words, differing by at most one character, from one word to the other
- Consider cold and warm: cold - cord - card - ward - warm -
- We will focus on 5 letter english words taken from Donalds Knuths compilation of 5,756 5-letter words
-