A. Assertions

This is a way of combining if and raise:

```python
if ...:
    raise ValueError("...")
assert x<y, "___"
assert x<y, "___"
```

Used in many places where you want to make sure that some condition is true before continuing. We will use it to check types of parameters:

```python
def insert_number_into_list(num, L):
    assert type(num) is float, "num must be a float!"
    assert type(L) is list, "L must be a list!"
    # the code of the function here
```

B. Recursion

Review

three conditions for a function to be recursive:
1. It must call itself within the body of code
2. There must be a base case and a recursive step; this is usually inside an if/else
3. In the recursive step, where the function calls itself, it must set up a smaller version of the same problem

Recursive sorts

We talked about mergesort()

How do you do linear_merge of two lists? Write it out:
with open(filename) as f:
    text = f.read()

March 10 midterm
To merge two sorted lists L1 and L2:

In English:
To sort a list of numbers called L:
   If the list is short (only 0 or 1 elements long) merely return the list unchanged.
   Otherwise, take off the first element, sort the rest of the list using this algorithm, and then insert the first element into the new sorted sublist.

To insert a number N into a sorted list L:
   If N is less than the first element of L, merely form a new list with N preceding the element of L and return that.
   Otherwise, take off the first element of L and save it, insert N into the sublist that consists of the original L sans the first element using this algorithm, and then when you get that sorted lists back, merely insert the original first element into the new list.

Problem 1:
5 into [6, 8, 12, 15] => [5, 6, 8, 12, 15]

Problem 2:
5 into [2, 4, 6, 8, 9] => [2] + insert 5 into [4, 6, 8, 9]

subproblem A: insert 5 into [4, 6, 8, 9]
5 into [4, 6, 8, 9] => [4] + insert 5 into [6, 8, 9]

subproblem B: insert 5 into [6, 8, 9]

this is really easy! Since 5 < 6, just prepend 5, and return [5, 6, 8, 9]

This is the value of subproblem B, so [4] + [5, 6, 8, 9] which gives [4, 5, 6, 8, 9] which is the value of subproblem A

[2] + [4, 5, 6, 8, 9] => [2, 4, 5, 6, 8, 9]

Let's write this in Python now!
def count_occurrences(x, L):
    if empty(L):
        return 0
    else:
        if head(L) == x:
            return 1 + count_occurrences(x, tail(L))
        else:
            return count_occurrences(x, tail(L))

def merge(L1, L2):
    if empty(L1):
        return L2
    elif empty(L2):
        return L1
    elif head(L1) <= head(L2):
        return insert(head(L1), merge(tail(L1), L2))
    else:
        return insert(head(L2), merge(L1, tail(L2)))
**Bubble sort**

```python
def insert_sorted(x, L):
    if len(L) == 0:
        return [x]
    if x < L[0]:
        return [x] + L
    else:
        return [L[0]] + insert_sorted(x, L[1:])

x = [2, 5, 7, 9, 15, 16, 17]
print(insert_sorted(20, x))

def sort(L):
    if len(L) < 2:
        return L
    return insert_sorted(L[0], sort(L[1:]))

print(sort([5, 2, 7, 4, 9, 10, 15, 8]))
```

Sorts done non-recursively are often difficult to understand.

```python
def bubble(L):
    for i in range(0, len(L)-1):
        if L[i] > L[i+1]:
            L[i], L[i+1] = L[i+1], L[i]

x = [5, 2, 7, 3, 8, 6, 9, 4]
bubble(x)
print(x)
```

# oops! Not sorted yet!
# bubble again!

def bubble_sort(L):
    for k in range(0, len(L)):
        bubble(L)

sort([5, 2, 4, 6, 8, 3])
insert_sorted(6, sort([2, 4, 6, 8, 3]))

[2, 3, 4, 6, 8]
Game-playing

Backtracking – all possible solutions

Maze threading – find your way out of a maze

Mathematics

multiplication – like exponentiation

log2

gcd

Lists

Tons of things here! The helper functions are very useful. See below:

```python
def head(L):
    return L[0]  # returns an element

def tail(L):
    return L[1:]  # returns a list (sublist)

def empty(L):
    return L == []

def insert(x, L):
    return [x] + L  # returns list

def append(x, L):
    return L + [x]
```

To count elements in a list:

```python
def get_length(L):
    if empty(L): return 0
    else: return 1 + get_length(tail(L))
```

Write a similar one: count how many times X occurs in a list

```python
def count_occurrence(X, L):
```
try

<table>
<thead>
<tr>
<th>call</th>
</tr>
</thead>
</table>

except

\[
\text{catch (Exception } e \text{)}
\]

\[
\text{raise}
\]

throw new Exception
if len(L) ≤ 2:
    return L

return insert_sorted(L[0], sort(L[1:]))

insert_sorted(x, L)
if len(L) = 0:
    return [x]
if x < L[0]:
    return [x] + L
else
    return insert_sorted([L[0],
                           L[0] + insert_sorted(x, L[1:])])
5 (4, 7) [4, sort (5, 7)]
5 (6, 7) [5, sort (6, 7)]
5 (7, 6) [5, sort (7, 6)]
5 (7, 4) [

inserted sorted 5

4 5, 7

if
\[2 \times 3! - 5\]
\[(2 + 3)! - 1\]
\[\frac{3}{3}!\]
\[2 \times 3!\]

\[2 \times 5\]
\[2 \times 5\]
\[2 \times (5+1)\]
\[2 \times \]

```python
def is_id(s):
    op < ~
    ~ > op
```

```python
a = 5;
```

```python
* filename
```