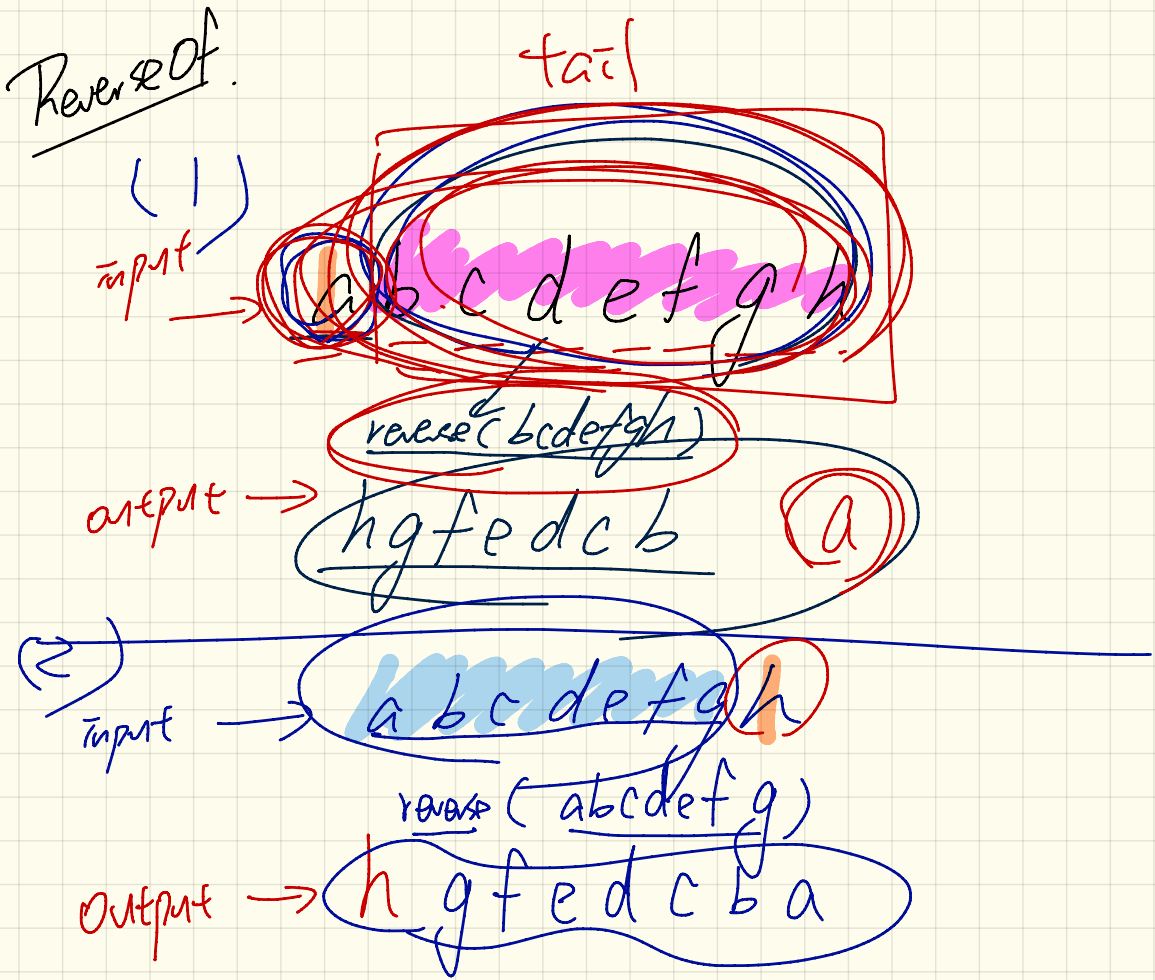
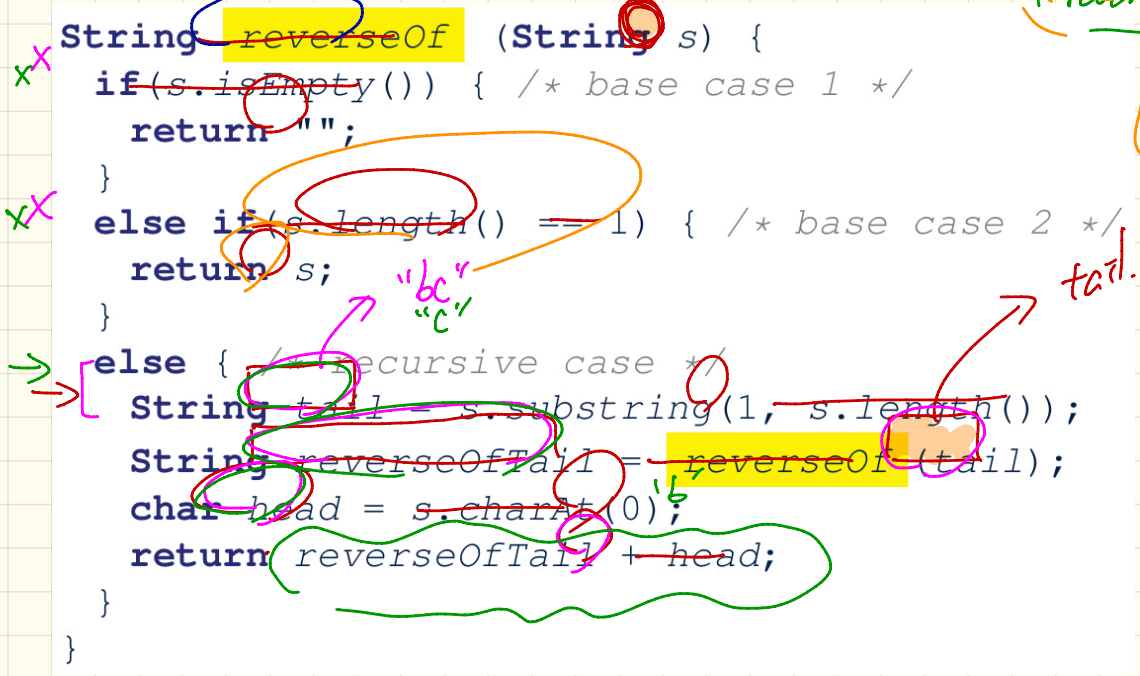
Monday Nov. 26

Lecture 22


Reverse of.
tail
input
a b c d e f g h
reverse(bcdefgh)
output
h g f e d c b
a
input
a b c d e f g h
reverse(abcdefg)
output
h g f e d c b a

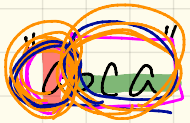
Reverse of a String



```
String reverseOf (String s) {
  if(s.isEmpty()) { /* base case 1 */
    return "";
  }
  else if(s.length() == 1) { /* base case 2 */
    return s;
  }
  else { /* recursive case */
    String tail = s.substring(1, s.length());
    String reverseOfTail = reverseOf(tail);
    char head = s.charAt(0);
    return reverseOfTail + head;
  }
}
```

Number of Occurrences

occ(abca)



a == a
1

occ("bca", 'a')
1

+ = 2

d == a
0

+ occ("bca", 'd')
0

+ = 0

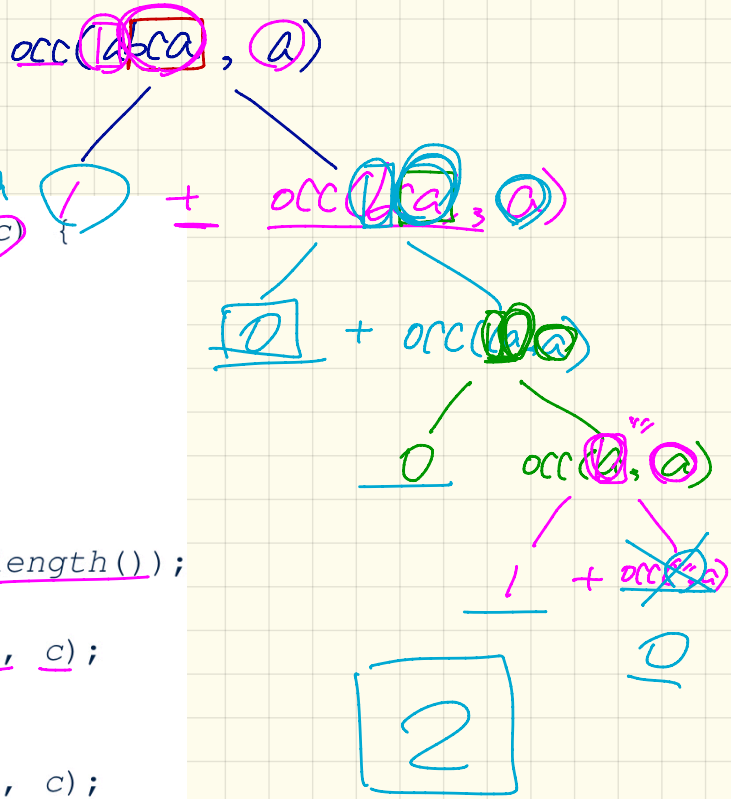
a == c
0

+ occ("bca", 'c')
1

+ = 1

Number of Occurrences

```
int occurrencesOf(String s, char c) {
  if(s.isEmpty()) {
    /* Base Case */
    return 0;
  }
  else {
    /* Recursive Case */
    char head = s.charAt(0);
    String tail = s.substring(1, s.length());
    if(head == c) {
      return 1 + occurrencesOf(tail, c);
    }
    else {
      return 0 + occurrencesOf(tail, c);
    }
  }
}
```



Recursion on Array: Passing a new array

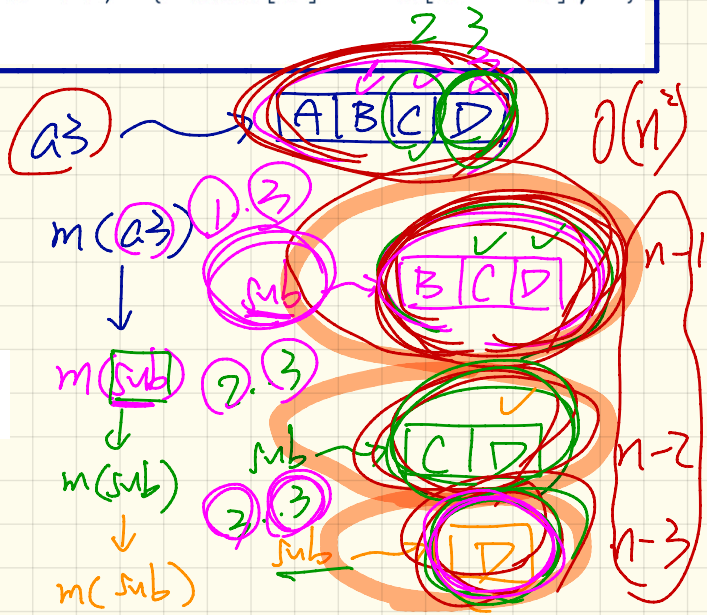
```
void m(int[] a) {
  if(a.length == 0) { /* base case */ }
  else if(a.length == 1) { /* base case */ }
  else {
    int[] sub = new int[a.length - 1];
    for(int i = 1; i < a.length; i ++) { sub[0] = a[i - 1]; }
    m(sub) } }
```

strictly smaller than a

Say a1 = {}, consider m(a1) ✓

Say a2 = {A}, consider m(a2) ✓

Say a3 = {A, B, C, D}, consider m(a3)



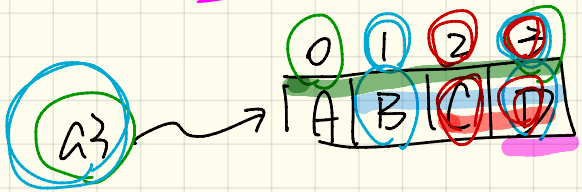
Recursion on Array: Passing an array reference

```
void m(int[] a, int from, int to) {
  if (from > to) { /* base case */ }
  else if (from == to) { /* base case */ }
  else { m(a, from + 1, to) } }
```

Say a1 = {}, consider m(a1)

Say a2 = {A}, consider m(a2)

Say a3 = {A, B, C, D}, consider m(a3)



m(a, 0, a1.length-1)

m(a2, 0, a2.length-1)

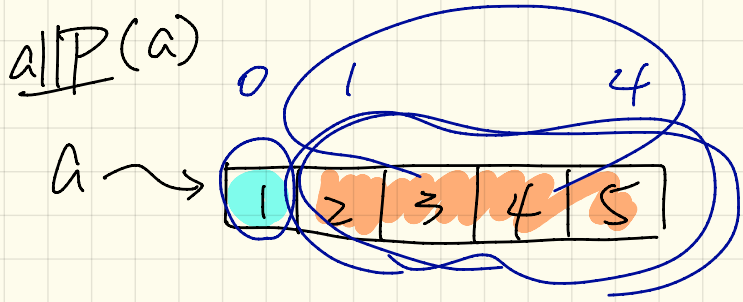
m(a3, 0, a3.length-1)

m(a3, 0, 3) ✓

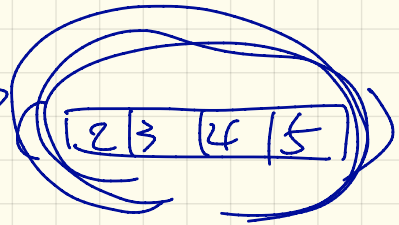
m(a3, 1, 3) ✓

m(a3, 2, 3) ✓

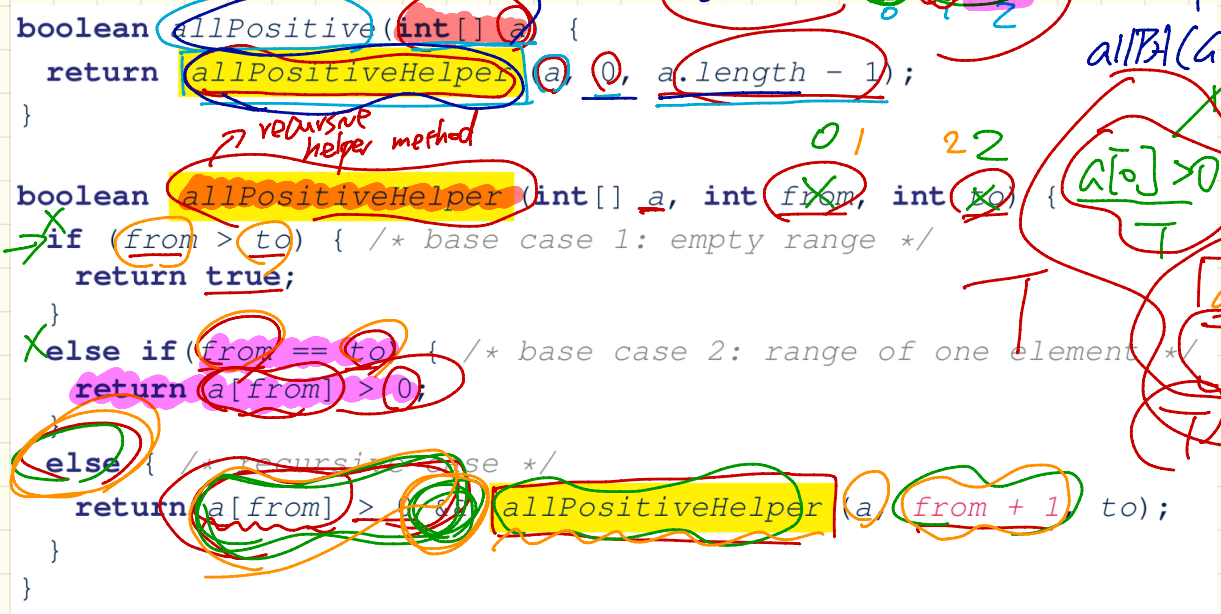
m(a3, 3, 3) ✓



allP(a) = a[0] > 0 && allP

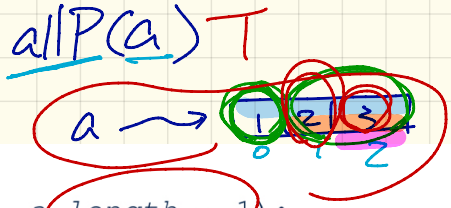


Are all numbers positive?

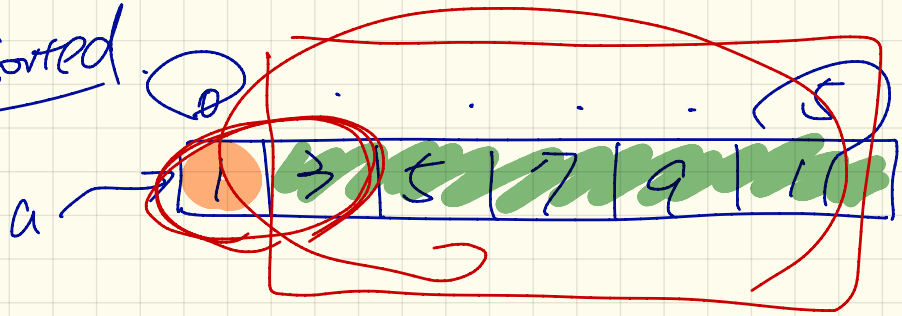


```
boolean allPositive(int[] a) {
  return allPositiveHelper(a, 0, a.length - 1);
}

boolean allPositiveHelper(int[] a, int from, int to) {
  if (from > to) { /* base case 1: empty range */
    return true;
  }
  else if (from == to) { /* base case 2: range of one element */
    return a[from] > 0;
  }
  else { /* recursive case */
    return a[from] > 0 && allPositiveHelper(a, from + 1, to);
  }
}
```

isSorted



isSorted (a, 0, 5)

= a[0] ≤ a[0+1] ___ isSorted (a, 0+1, 5)