Recursion Notes — Day 2

AP Computer Science A
[AI generated from my outline]

1. Counting Divisions: Floor of Log_2

How many times can we divide n by 2 before reaching 0?

```
public int logBase2(int n) {
    if (n < 1) return 0;
    return 1 + logBase2(n / 2);
}
```

Trace logBase2(16):

```
logBase2(16) = 1 + logBase2(8)
             = 1 + 1 + logBase2(4)
             = 1 + 1 + 1 + logBase2(2)
             = 1 + 1 + 1 + 1 + logBase2(1)
             = 1 + 1 + 1 + 1 + 1 + logBase2(0)
             = 1 + 1 + 1 + 1 + 1 + 0 = 5
```

Pattern: Each division by 2 adds 1 to the count. This computes floor($\log_2$(n)) + 1 for n ≥ 1.

2. String Length via Recursion

Count characters by peeling off one at a time:

```
public int length(String s) {
    if (s.equals("")) return 0;
    return 1 + length(s.substring(1));
}
```

Trace length("cat"):

```
length("cat") = 1 + length("at")
              = 1 + 1 + length("t")
              = 1 + 1 + 1 + length("")
              = 1 + 1 + 1 + 0 = 3
```

Pattern: Each recursive call removes one character and adds 1. Base case is the empty string.

3. Building Strings: Wrapping

Build a string by wrapping characters around a recursive result:

```
public String wrap(int n) {
    if (n <= 0) return "";
    return "a" + wrap(n - 1) + "b";
}
```

Trace wrap(3):

```
wrap(3) = "a" + wrap(2) + "b"
        = "a" + ("a" + wrap(1) + "b") + "b"
```

```
= "a" + ("a" + ("a" + wrap(0) + "b") + "b") + "b"
= "a" + ("a" + ("a" + "" + "b") + "b") + "b"
= "a" + ("a" + "ab" + "b") + "b"
= "a" + "aabb" + "b"
= "aaabbb"
```

Pattern: Adding before AND after the recursive call creates symmetric/nested structures.

4. Searching: Contains Character

Return true if the string contains at least one 'm':

```
public boolean containsM(String s) {
    if (s.length() == 1) return s.charAt(0) == 'm';
    return s.charAt(0) == 'm' || containsM(s.substring(1));
}
```

Trace containsM("camp"):

```
containsM("camp"): 'c'=='m'? No → || containsM("amp")
containsM("amp"): 'a'=='m'? No → || containsM("mp")
containsM("mp"): 'm'=='m'? Yes → true (short-circuit)
```

Logic: Base case is a single character — just check it directly. Recursive case: true if first char is 'm' OR the rest contains 'm'.

5. Stricter Check: Contains ONLY 'm'

Return true only if EVERY character is 'm':

```
public boolean onlyM(String s) {
    if (s.length() == 1) return s.charAt(0) == 'm';
    return s.charAt(0) == 'm' && onlyM(s.substring(1));
}
```

Trace onlyM("mmm"):

```
onlyM("mmm"): 'm'=='m'? Yes → && onlyM("mm")
onlyM("mm"): 'm'=='m'? Yes → && onlyM("m")
onlyM("m"): length==1, 'm'=='m'? → true
Unwind: true && true && true = true
```

Trace onlyM("mom"):

```
onlyM("mom"): 'm'=='m'? Yes → && onlyM("om")
onlyM("om"): 'o'=='m'? No → false (short-circuit)
```

Key difference: Use && instead of ||. "Contains" needs only ONE match (||). "Only" requires ALL to match (&&).

6. Challenge: Random Palindrome Builder

Write a function that creates a palindrome of length 2m using random 'a' and 'b' characters.

Key insight: A palindrome reads the same forward and backward. So we pick a random character and put it on BOTH ends.

```
public String randomPalindrome(int m) {
    if (m <= 0) return "";
    String c = (Math.random() < 0.5) ? "a" : "b";
    return c + randomPalindrome(m - 1) + c;
}
```

Example trace of randomPalindrome(3) (assuming random picks a, b, a):

```
randomPalindrome(3): c="a" → "a" + randomPalindrome(2) + "a"
randomPalindrome(2): c="b" → "b" + randomPalindrome(1) + "b"
randomPalindrome(1): c="a" → "a" + randomPalindrome(0) + "a"
randomPalindrome(0): ""
Unwind: "aa", then "baab", then "abaaba"
```

Result: "abaaba" — a palindrome of length 6 = 2×3 ✓

Why it works: Each level wraps the same character on both sides, guaranteeing symmetry.

Summary

✓ Recursion with n/2 → logarithmic (counting divisions)

✓ Recursion with substring(1) → linear traversal of string

✓ Returning "a" + f(...) + "b" → builds symmetric/nested structures

✓ "Contains" logic: return true early when found, false at empty

✓ "Only" logic: return false early when violation found, true at empty

✓ To build a palindrome: wrap the SAME character on both sides