Quiz Prep Session - Week 12

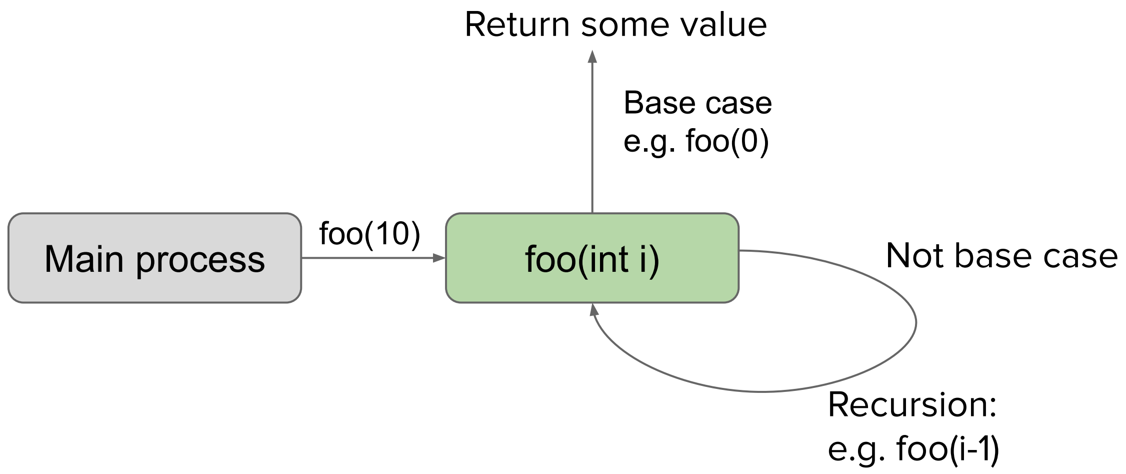
Recursion and Tree

EMP Session

1. 11/12 - Recursion and Tree
    a. https://cs199emp.netlify.app/dist/2020-11-12.html

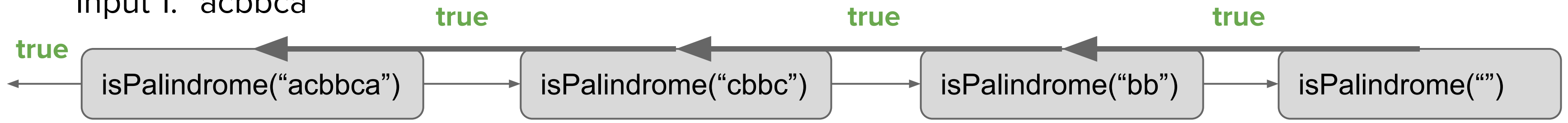
2. 11/17 - Recursion and Tree p2
    a. https://cs199emp.netlify.app/dist/2020-11-17.html

Recursion


Return some value
Base case
e.g. foo(0)
Main process
foo(10)
foo(int i)
Not base case
Recursion:
e.g. foo(i-1)

Recursion - isPalindrome (String in)

Input 1: “acbbca”



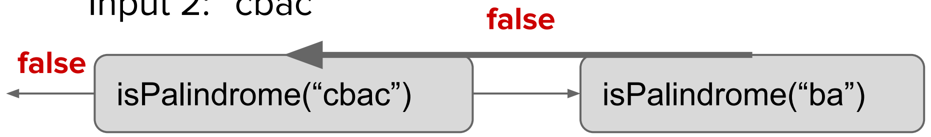
check the characters on both ends of the string.
> “a” = “a” -> **true, look at substring:**
> “acbbca” -> “cbbc”

> “c” = “c” -> **true, look at substring:**
> “cbbc” -> “bb”

> “b” = “b” -> **true, look at substring:**
> “bb” -> “” # empty string

Input string length <= 1,
So this is a base case:
Return **true**

Input 2: “cbac”



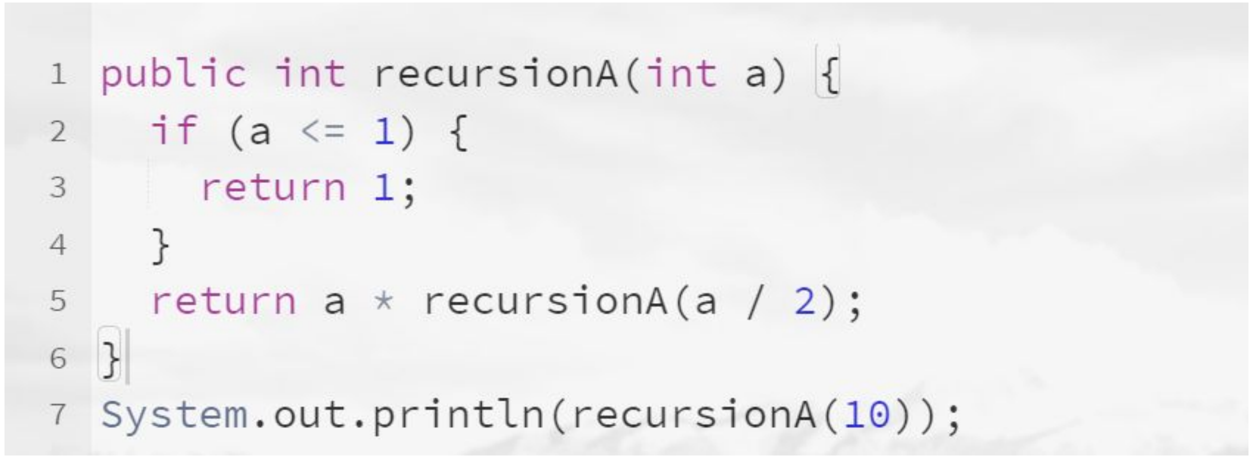
> “c” = “c” -> **true, look at substring:**
> “cbac” -> “ba”

> “b” != “a” -> return **false**

Recursion - Practices

Practice 1.a

How many recursion calls are made in the following codes (including the initial call)?

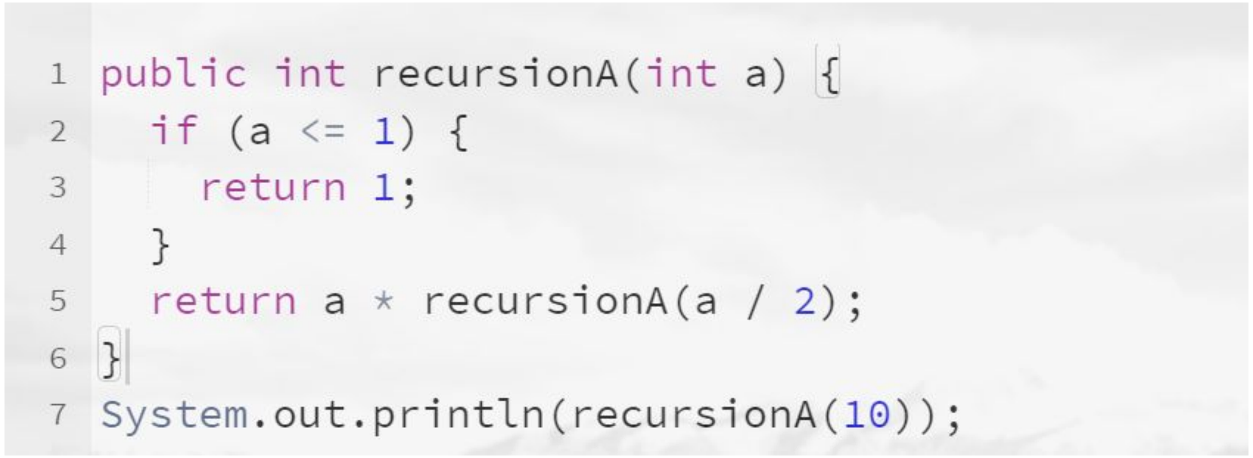


```java
public int recursionA(int a) {
  if (a <= 1) {
    return 1;
  }
  return a * recursionA(a / 2);
}
System.out.println(recursionA(10));
```

Recursion - Practices

Practice 1.b

What will be printed out?



```
public int recursionA(int a) {
  if (a <= 1) {
    return 1;
  }
  return a * recursionA(a / 2);
}
System.out.println(recursionA(10));
```

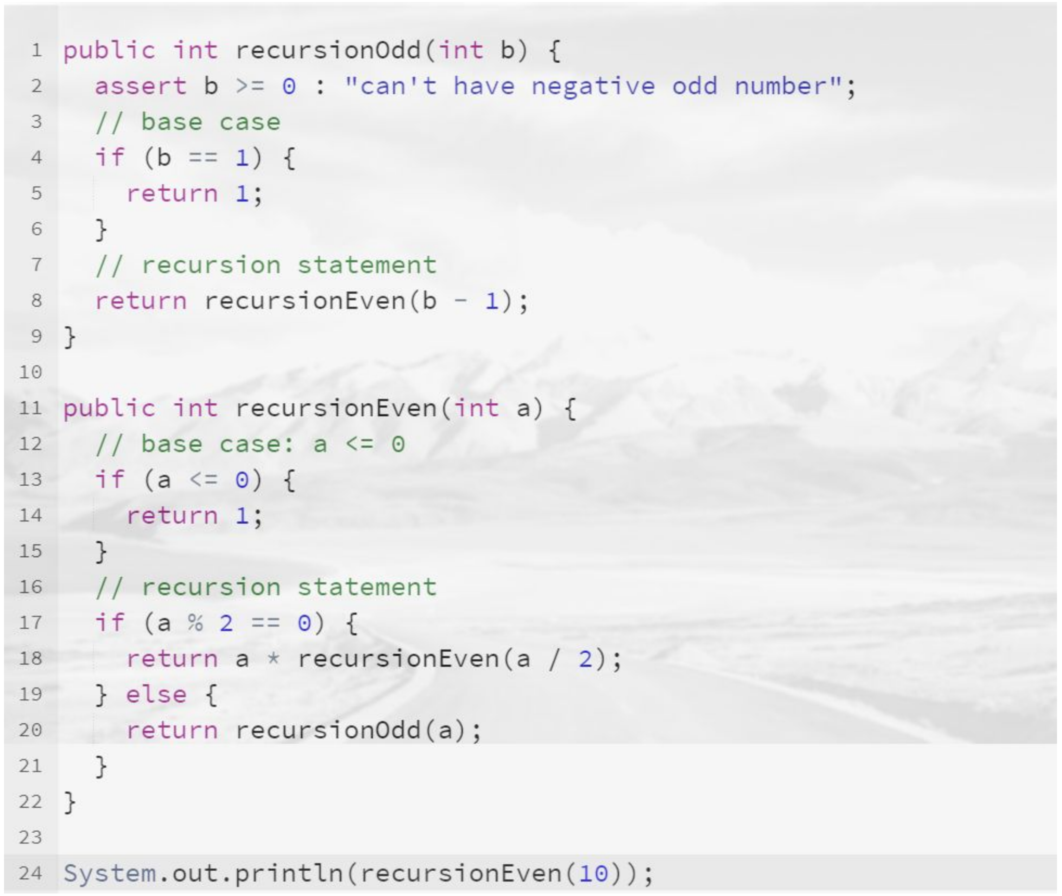
Recursion - Practices

A little bit harder!
Practice 1 - Special Edition

How many times are *recursionEven()* called?

How many times are *recursiveOdd()* called?

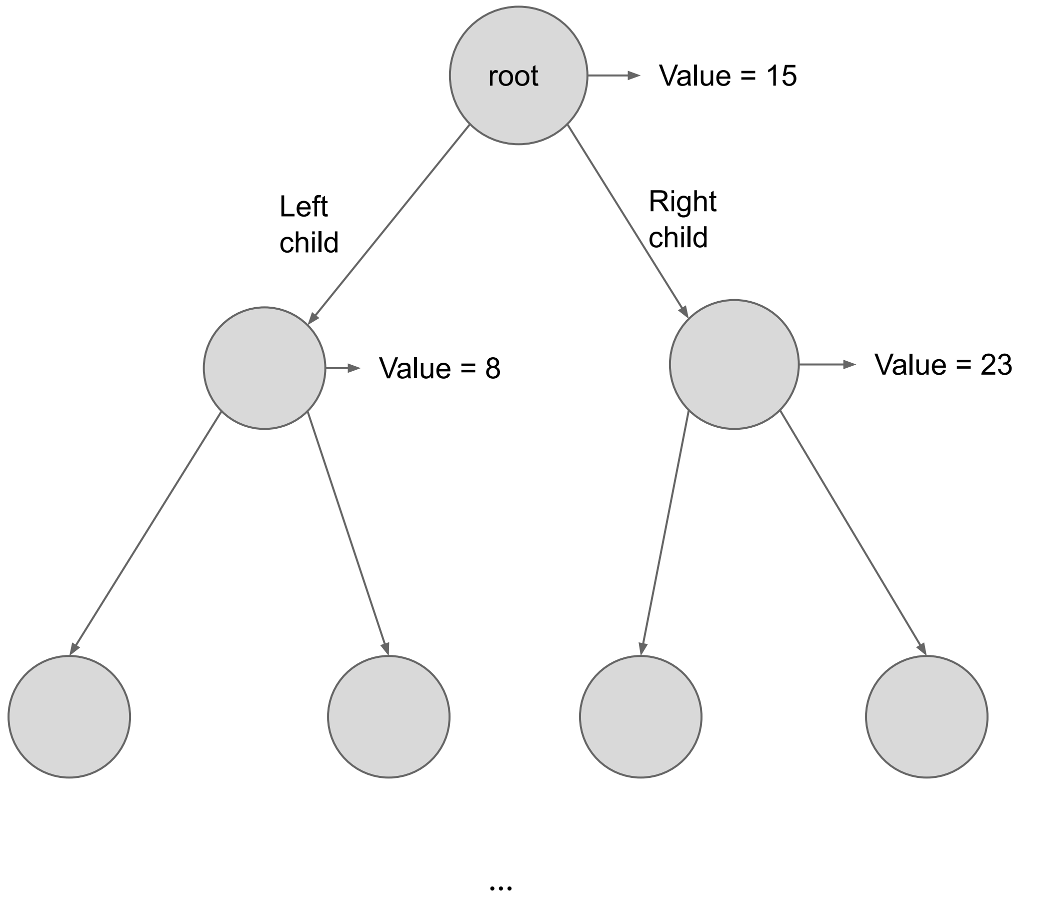
What will be printed out?



```
public int recursionOdd(int b) {
  assert b >= 0 : "can't have negative odd number";
  // base case
  if (b == 1) {
    return 1;
  }
  // recursion statement
  return recursionEven(b - 1);
}

public int recursionEven(int a) {
  // base case: a <= 0
  if (a <= 0) {
    return 1;
  }
  // recursion statement
  if (a % 2 == 0) {
    return a * recursionEven(a / 2);
  } else {
    return recursionOdd(a);
  }
}

System.out.println(recursionEven(10));
```

Trees

A binary tree structure



Depth h = 0

h = 1

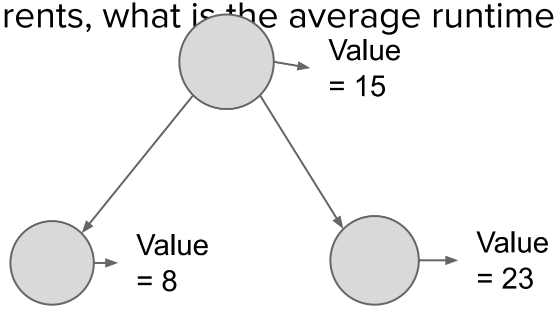
h = 2

Trees

Important aspects regarding a BinaryTree with **N** nodes and **H** depths:

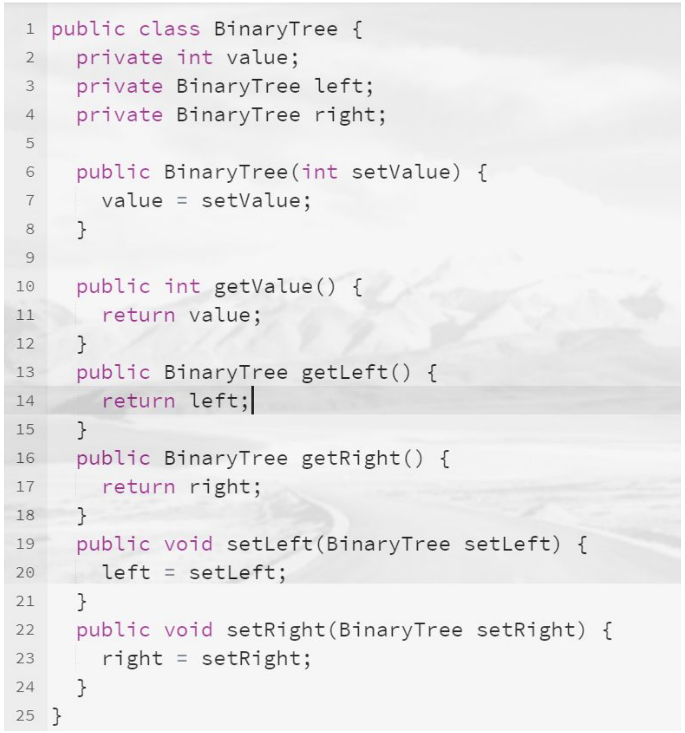
1. (Tree size) If each depth **h** in the tree is totally filled, what is the size **N** of the tree represented by **H**?

2. (Binary Search Tree) Assume all left child nodes have **smaller values than** their parents, and all right child nodes have **larger values than** their parents, what is the average runtime for searching for a value in the tree?



Trees - Practices

Given the following definition of a BinaryTree class:

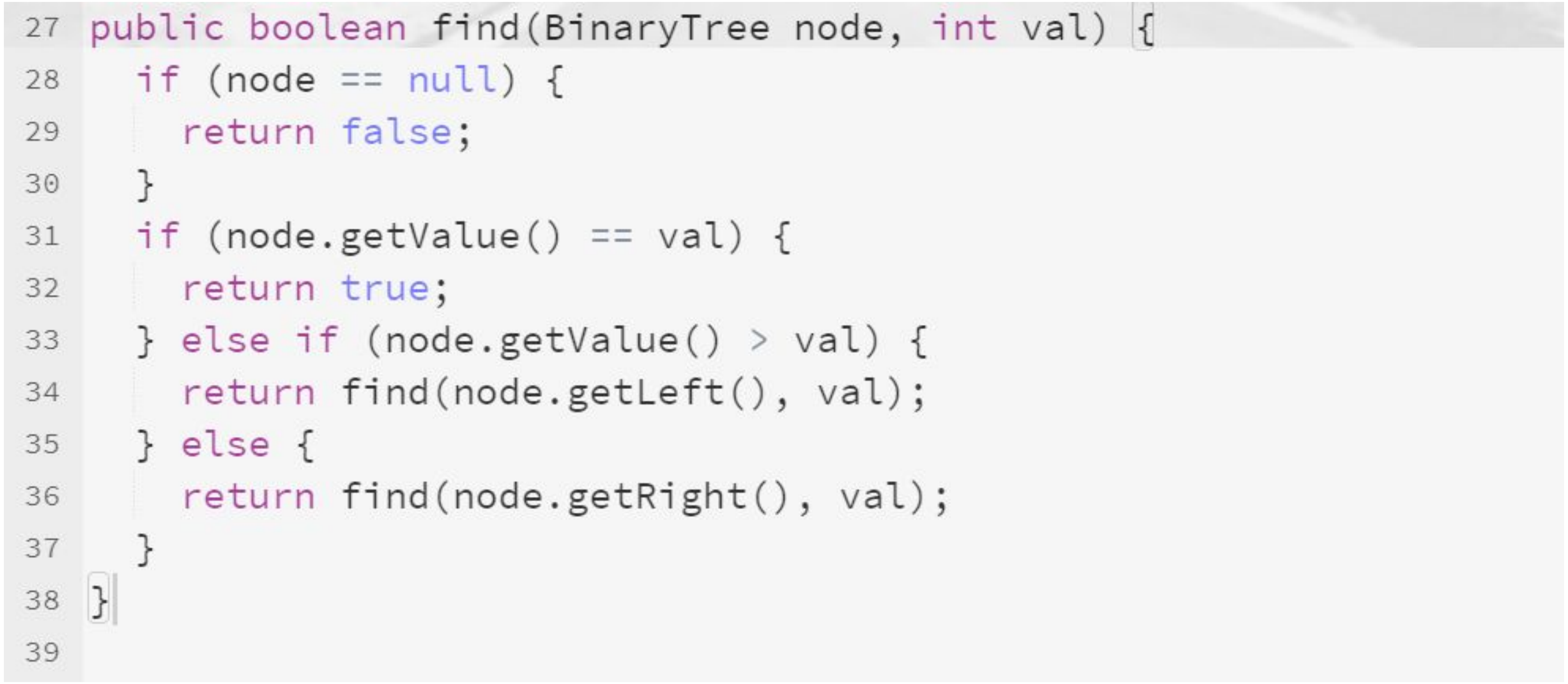


```java
public class BinaryTree {
  private int value;
  private BinaryTree left;
  private BinaryTree right;

  public BinaryTree(int setValue) {
    value = setValue;
  }

  public int getValue() {
    return value;
  }
  public BinaryTree getLeft() {
    return left;
  }
  public BinaryTree getRight() {
    return right;
  }
  public void setLeft(BinaryTree setLeft) {
    left = setLeft;
  }
  public void setRight(BinaryTree setRight) {
    right = setRight;
  }
}
```

Write a method to search for a certain value from the root. If the value exists in the tree (any node), return **true**, otherwise return **false**. We also have the following assumptions:

1. Left child nodes have smaller values than their parents;
2. Right child nodes have larger values than their parents.

Trees - Practice Solution



```java
public boolean find(BinaryTree node, int val) {
  if (node == null) {
    return false;
  }
  if (node.getValue() == val) {
    return true;
  } else if (node.getValue() > val) {
    return find(node.getLeft(), val);
  } else {
    return find(node.getRight(), val);
  }
}
```