**Problem 1: 69. Sqrt(x)**

Implement int sqrt(int x).

Compute and return the square root of *x*, where *x* is guaranteed to be a non-negative integer.

Since the return type is an integer, the decimal digits are truncated and only the integer part of the result is returned.

**Example 1:**

**Input:** 4

**Output:** 2

**Example 2:**

**Input:** 8

**Output:** 2

**Explanation:** The square root of 8 is 2.82842..., and since

  the decimal part is truncated, 2 is returned.

Solution:

class Solution {

public int mySqrt(int x) {

if (x == 0 || x == 1) return x;

// Binary Search

int left = 0, right = x;

while (left < right) {

// mid = (left + right) / 2 can overflow if right > Integer.MAX\_VALUE - left

int mid = left + (right - left) / 2;

// same thing here , mid \* mid > x can overflow. replace by mid > x / mid

if (mid > x / mid) {

right = mid - 1;

} else {

left = mid + 1;

// if mid \* mid < x but (mid + 1) \* (mid + 1) > x then mid was the right answer

if (left > x / left) {

return mid;

}

}

}

return left;

}

}

**Problem 2: 70. Climbing Stairs**

You are climbing a stair case. It takes n steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Note:** Given n will be a positive integer.

**Example 1:**

**Input:** 2

**Output:** 2

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

**Example 2:**

**Input:** 3

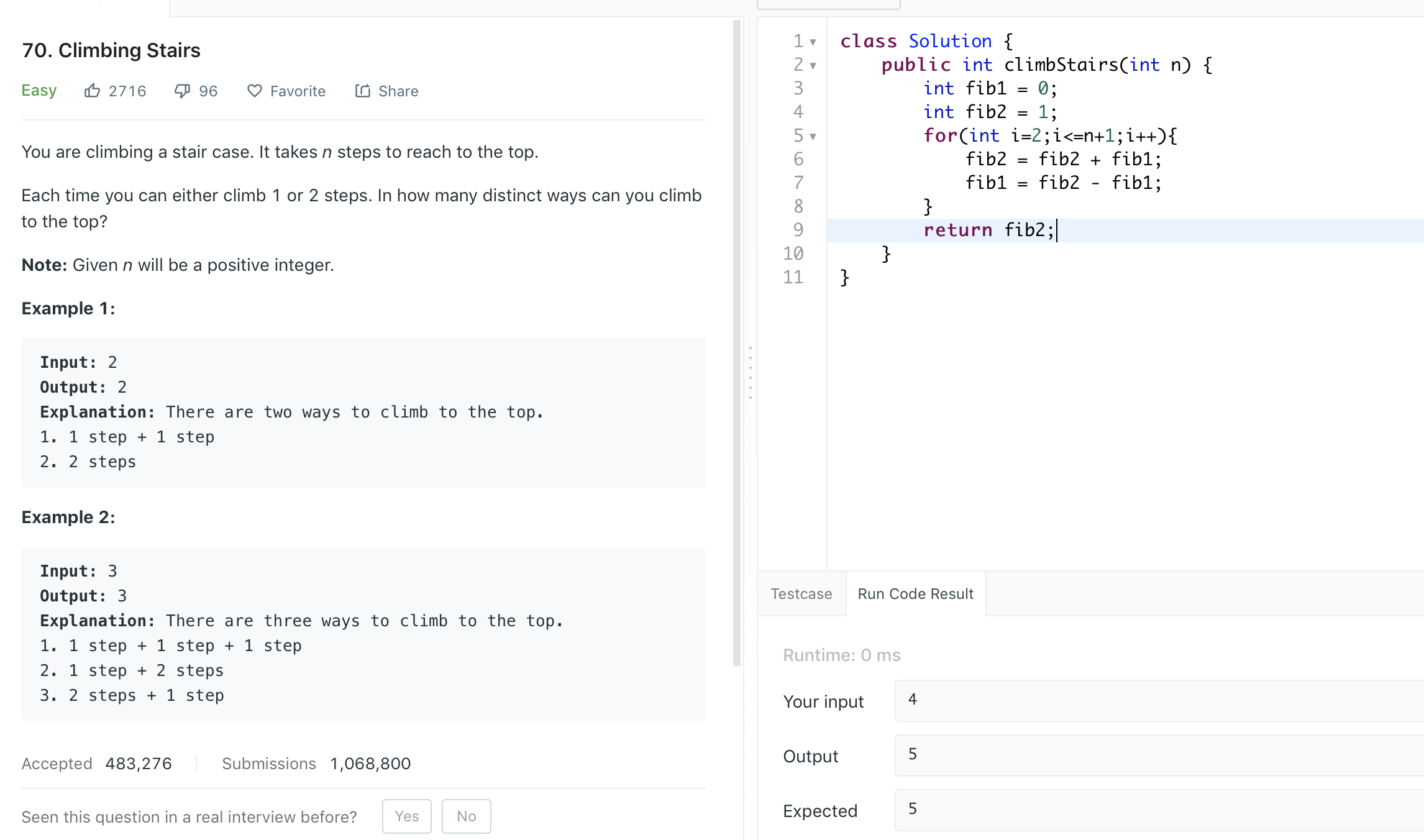
**Output:** 3

**Explanation:** There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step



**Problem 3: 88. Merge Sorted Array**

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

**Note:**

* The number of elements initialized in *nums1* and *nums2* are *m* and *n*respectively.
* You may assume that *nums1* has enough space (size that is greater or equal to *m* + *n*) to hold additional elements from *nums2*.

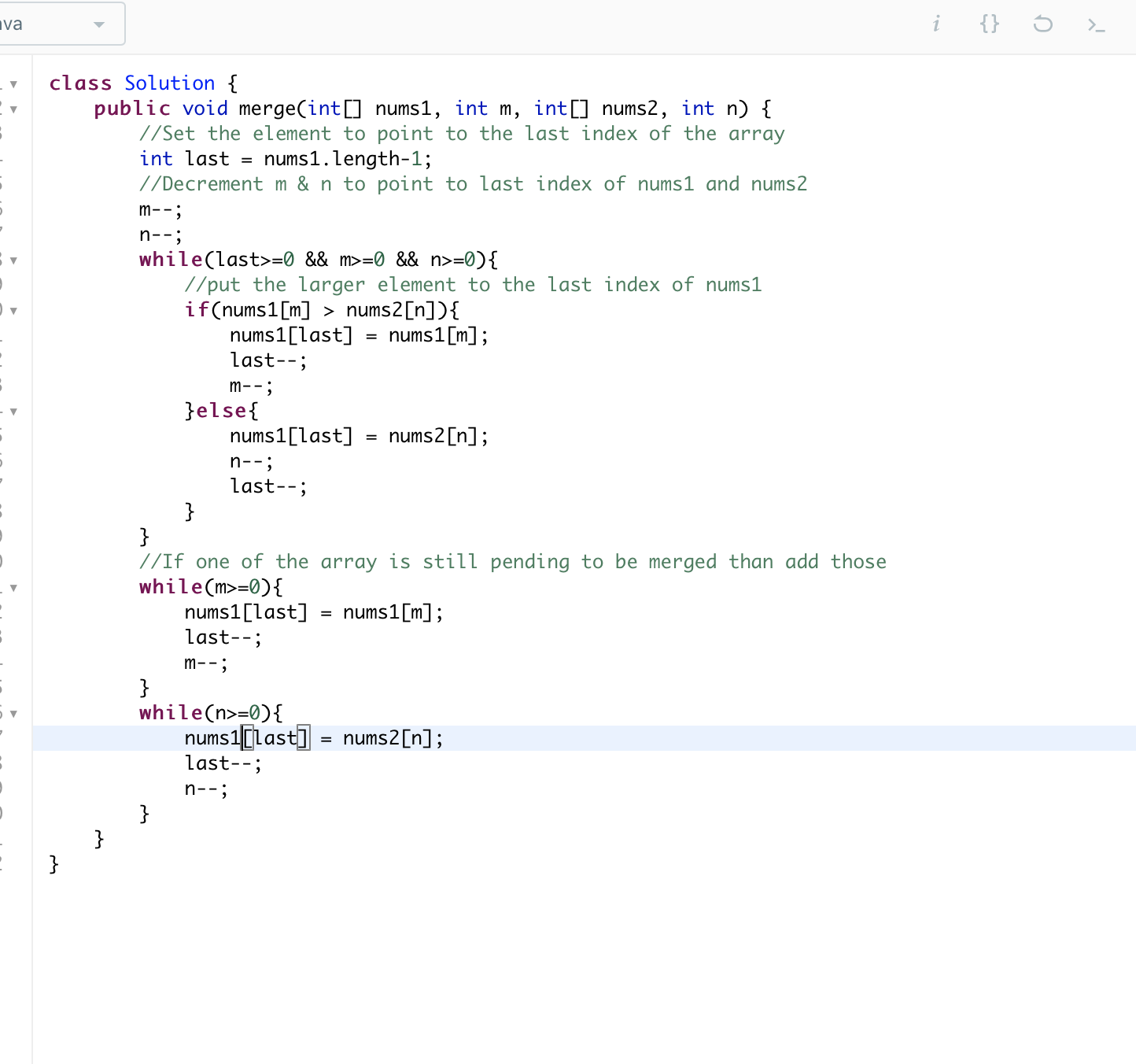
**Example:**

**Input:**

nums1 = [1,2,3,0,0,0], m = 3

nums2 = [2,5,6], n = 3

**Output:** [1,2,2,3,5,6]



Problem 4: **100. Same Tree**

Given two binary trees, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

**Example 1:**

**Input:** 1 1

/ \ / \

2 3 2 3

[1,2,3], [1,2,3]

**Output:** true

**Example 2:**

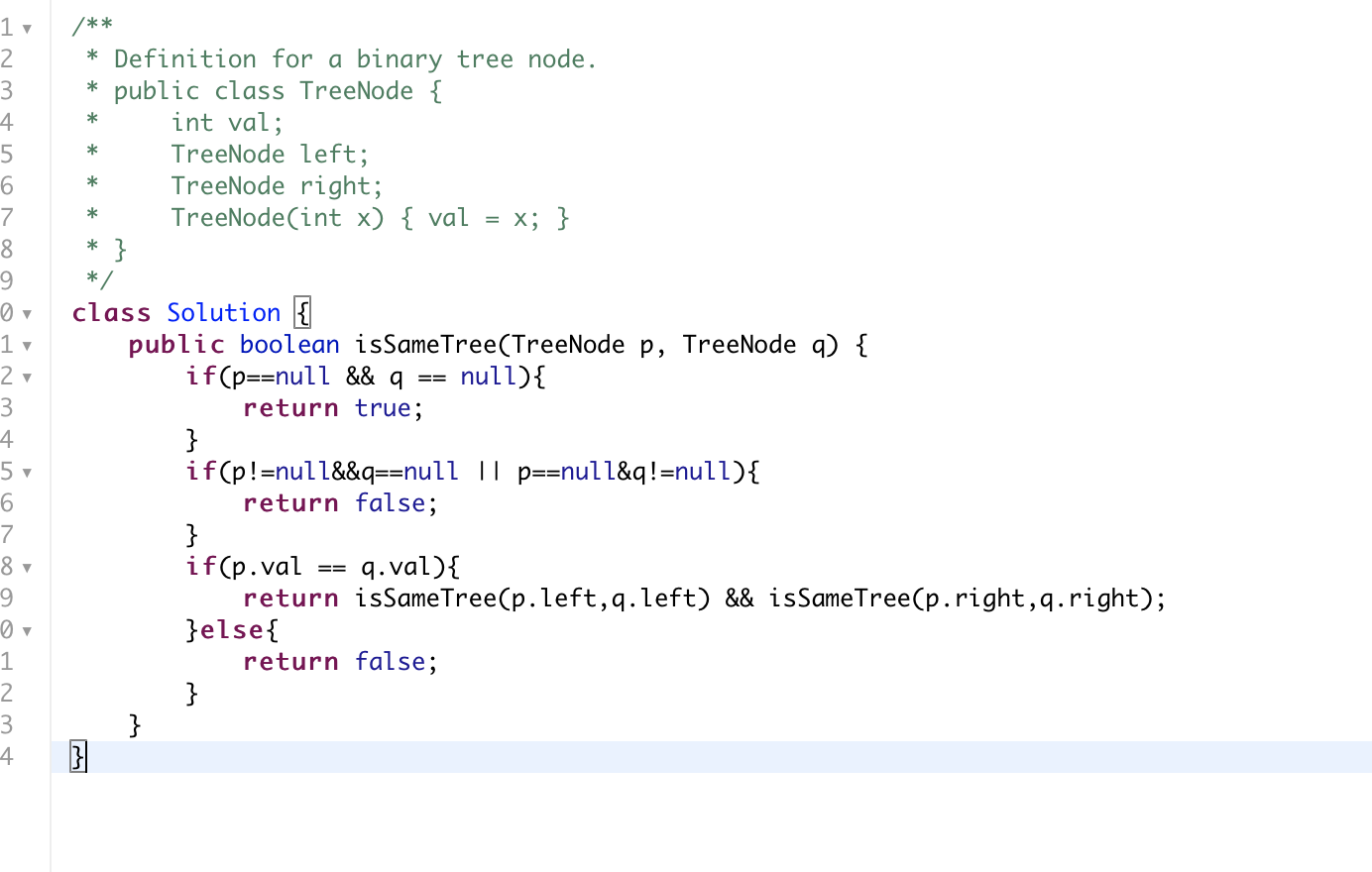
**Input:** 1 1

/ \

2 2

[1,2], [1,null,2]

**Output:** false



Problem 5: **101. Symmetric Tree**

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree [1,2,2,3,4,4,3] is symmetric:

1

/ \

2 2

/ \ / \

3 4 4 3

But the following [1,2,2,null,3,null,3] is not:

1

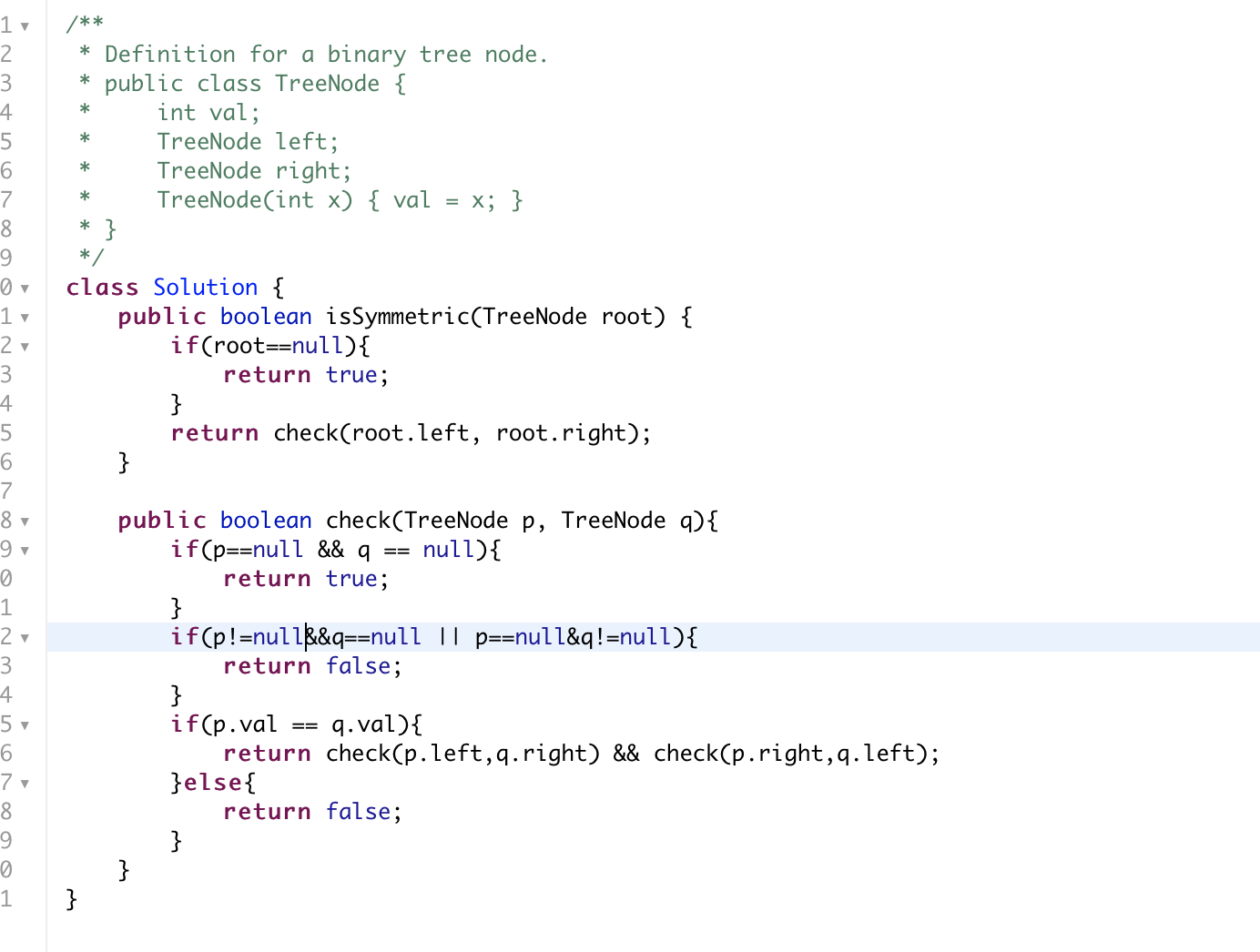
/ \

2 2

\ \

3 3

**Solution is similar to same tree problem. Just instead of recursing left or right of both we recurse on left of one and right of other and vice versa.**



Problem 6: **6. ZigZag Conversion**

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want to display this pattern in a fixed font for better legibility)

P A H N

A P L S I I G

Y I R

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows:

string convert(string s, int numRows);

**Example 1:**

**Input:** s = "PAYPALISHIRING", numRows = 3

**Output:** "PAHNAPLSIIGYIR"

**Example 2:**

**Input:** s = "PAYPALISHIRING", numRows = 4

**Output:** "PINALSIGYAHRPI"

**Explanation:**

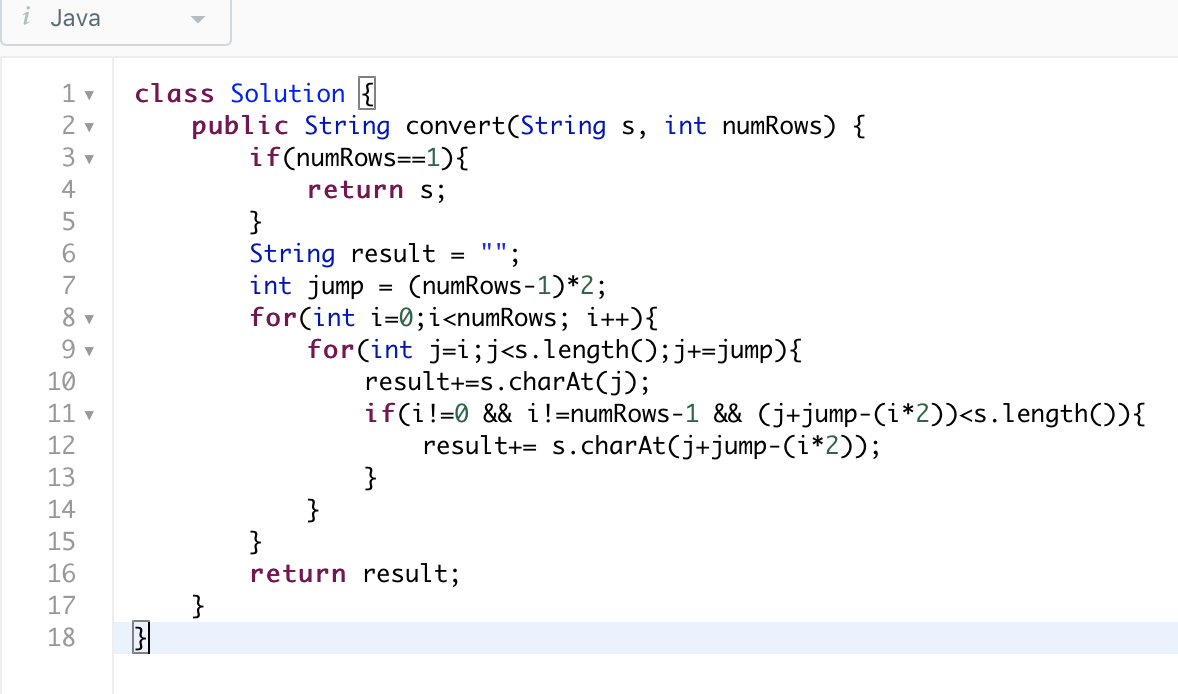
P I N

A L S I G

Y A H R

P I

Solution:



Problem 7:

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

**Note:** A leaf is a node with no children.

**Example:**

Given binary tree [3,9,20,null,null,15,7],

3

/ \

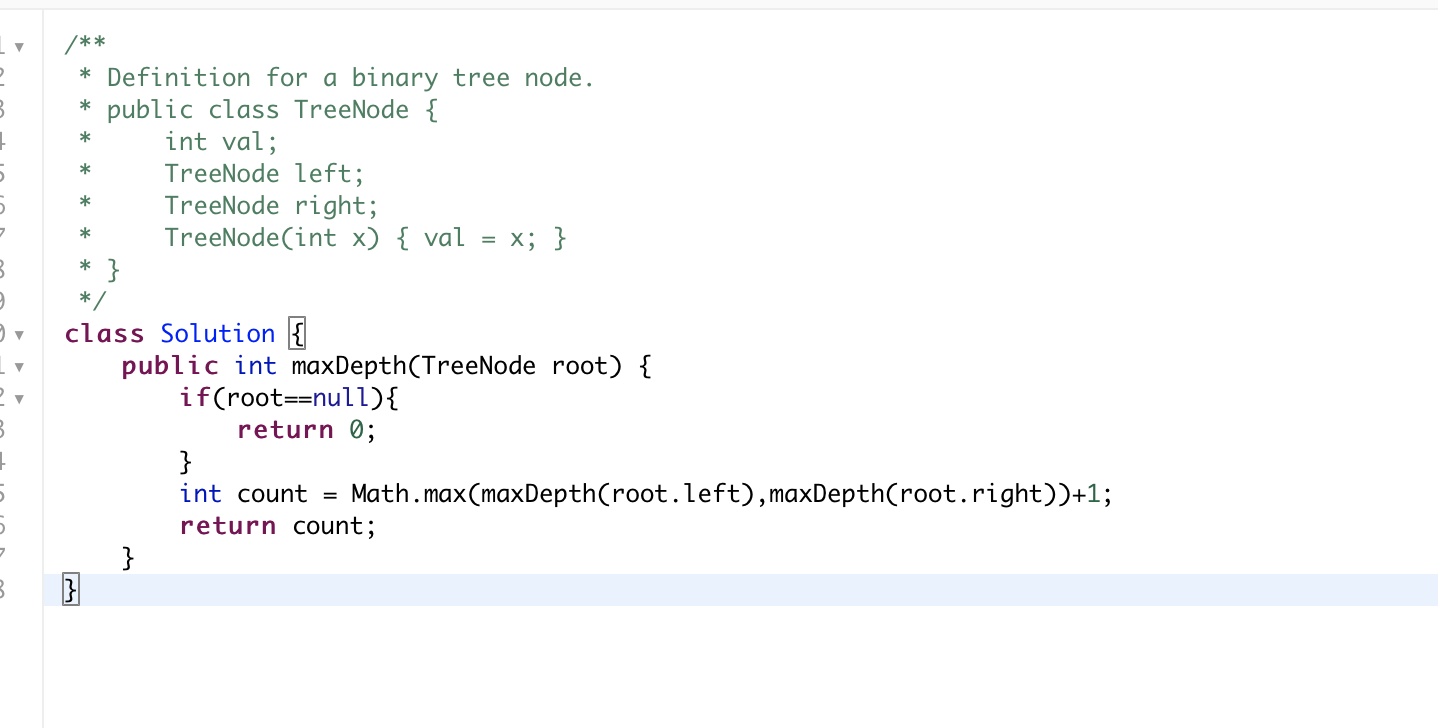
9 20

/ \

15 7

return its depth = 3.

Solution:



Problem 8: **19. Remove Nth Node From End of List**

Given a linked list, remove the *n*-th node from the end of list and return its head.

**Example:**

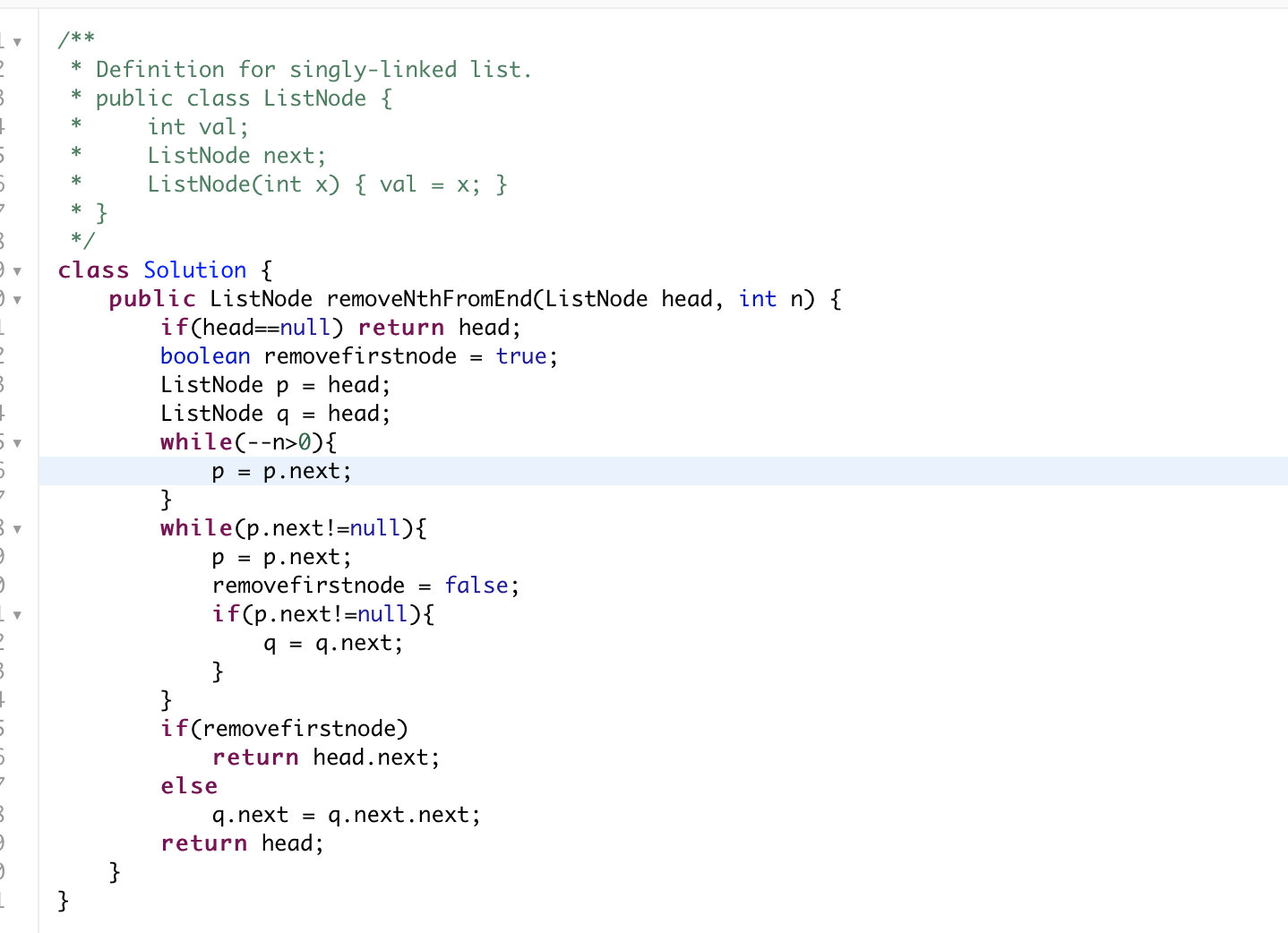
Given linked list: **1->2->3->4->5**, and ***n* = 2**.

After removing the second node from the end, the linked list becomes **1->2->3->5**.

**Note:**

Given *n* will always be valid.

Solution:



Problem 9: L