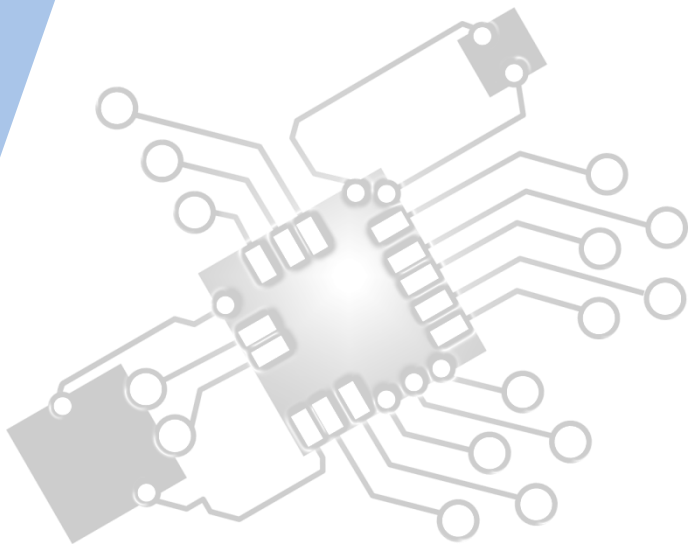
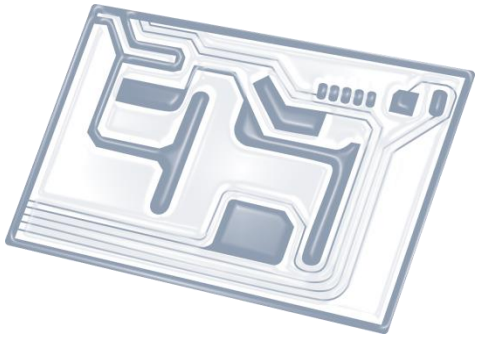
Abstract Data Structures

IB Computer Science



Content developed by
Dartford Grammar School
Computer Science Department

HL Topics 1-7, D1-4



1: System design

2: Computer Organisation

3: Networks

4: Computational thinking

5: Abstract data structures

6: Resource management

7: Control

D: OOP

HL only 5 Overview

Thinking recursively
5.1.1 Identify a situation that requires the use of recursive thinking
5.1.2 Identify recursive thinking in a specified problem solution
5.1.3 Trace a recursive algorithm to express a solution to a problem
Abstract data structures
5.1.4 Describe the characteristics of a two-dimensional array
5.1.5 Construct algorithms using two-dimensional arrays
5.1.6 Describe the characteristics and applications of a stack
5.1.7 Construct algorithms using the access methods of a stack
5.1.8 Describe the characteristics and applications of a queue
5.1.9 Construct algorithms using the access methods of a queue
5.1.10 Explain the use of arrays as static stacks and queues
Linked lists
5.1.11 Describe the features and characteristics of a dynamic data structure
5.1.12 Describe how linked lists operate logically
5.1.13 Sketch linked lists (single, double and circular)
Trees
5.1.14 Describe how trees operate logically (both binary and non-binary)
5.1.15 Define the terms: parent, left-child, right-child, subtree, root and leaf
5.1.16 State the result of inorder, postorder and preorder tree traversal
5.1.17 Sketch binary trees
Applications
5.1.18 Define the term dynamic data structure
5.1.19 Compare the use of static and dynamic data structures
5.1.20 Suggest a suitable structure for a given situation

1: System design

2: Computer Organisation

3: Networks

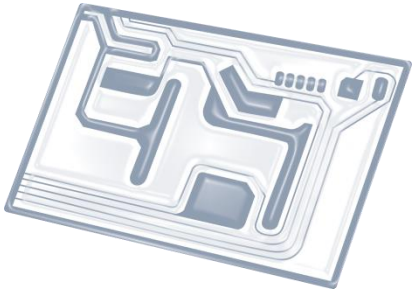
4: Computational thinking

5: Abstract data structures

6: Resource management

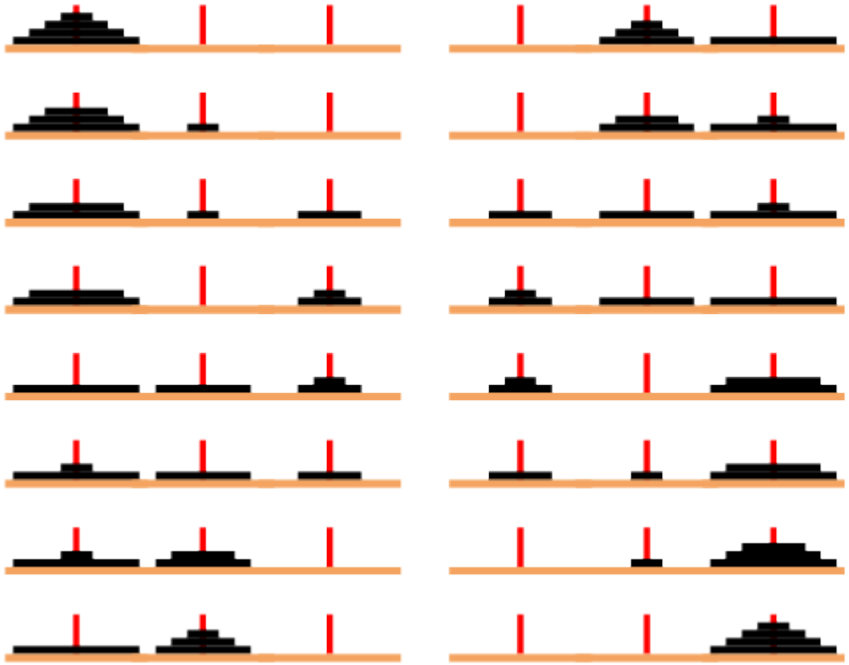
7: Control

D: OOP



Topic 5.1.2

Identify **recursive thinking** in a specified problem solution

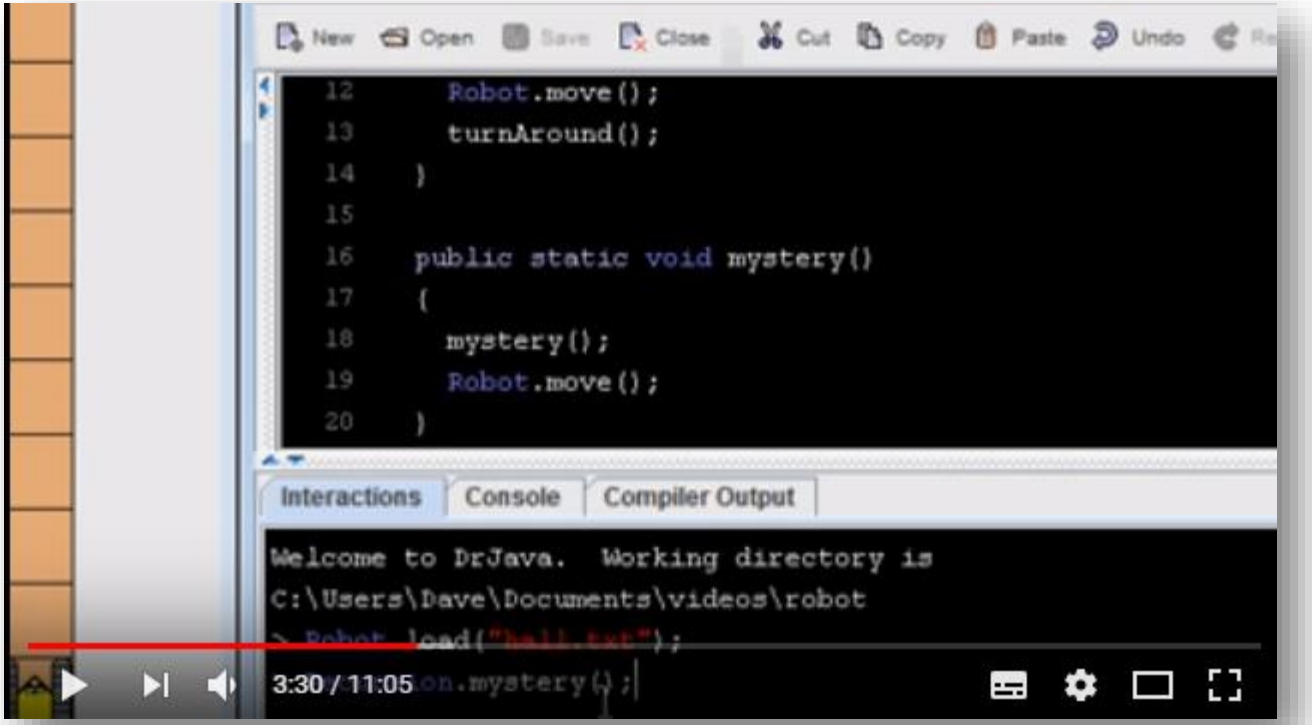


Exam note!

This topic should really be studied in both **pseudo code** (**Paper 1**) and **Java** (**Paper 2**) as it links with **topic D.4**.

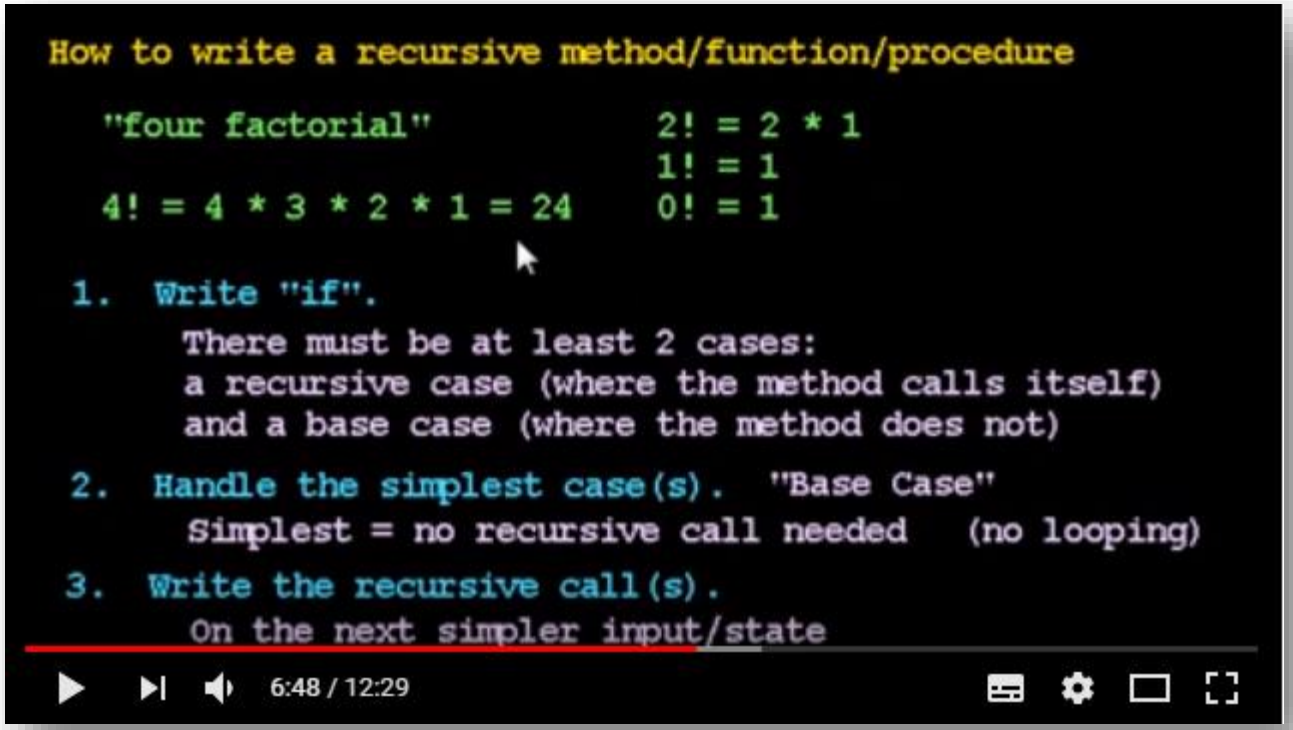
Students can expect both **algorithmic** and more **theory based questions** from this topic; answers could be a written paragraph or writing a pseudo code/Java method.

Video: Introducing Recursion



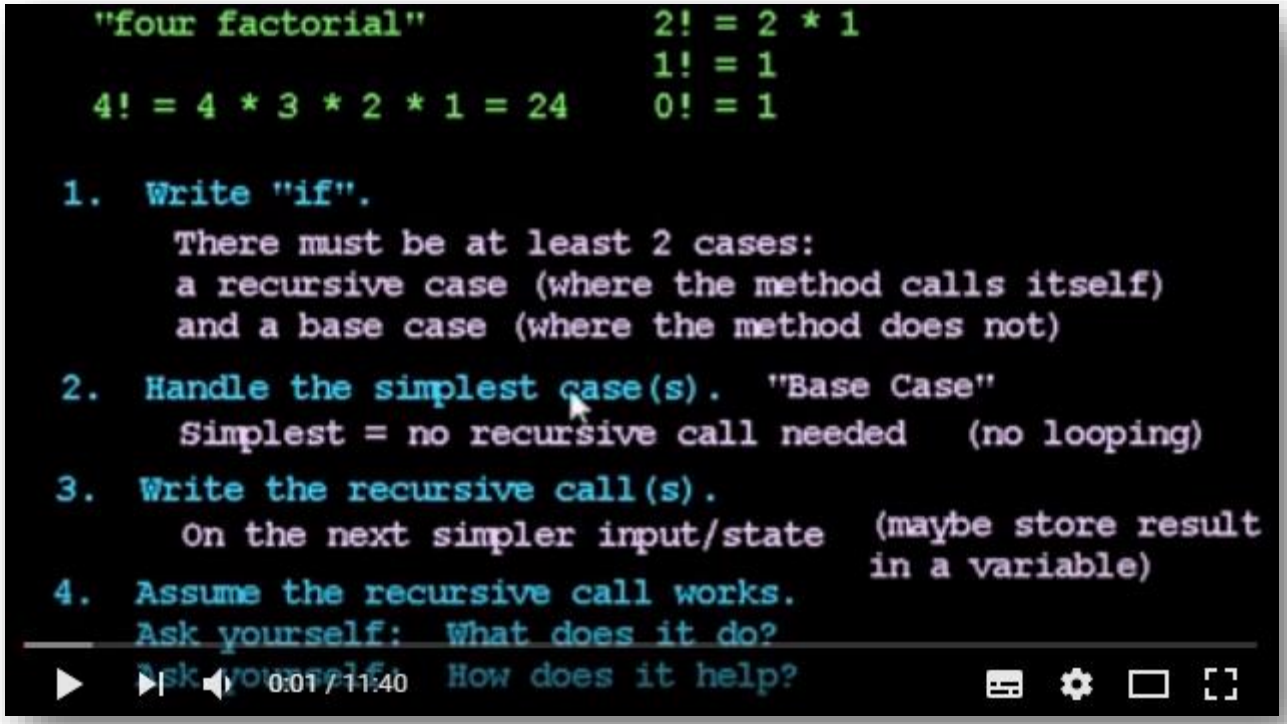
Link (YouTube): https://youtu.be/to7tAmiZ_Ic

Video: How to write a recursive method



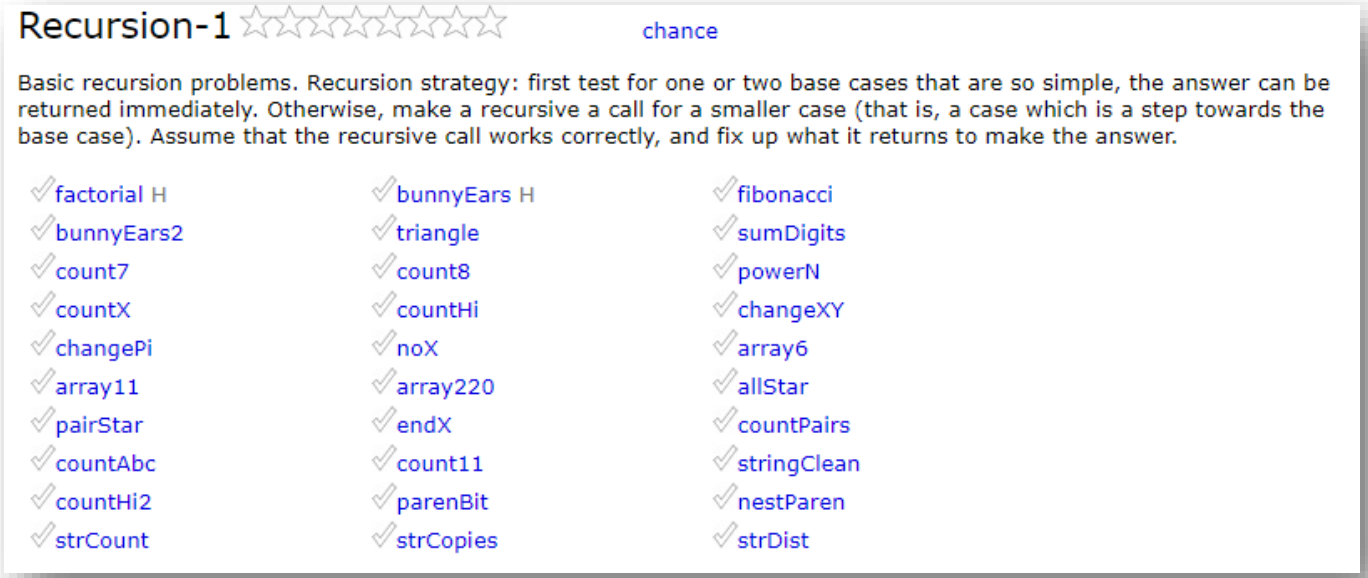
Link (YouTube): https://youtu.be/MyzFdthuUcA

Video: How recursion works



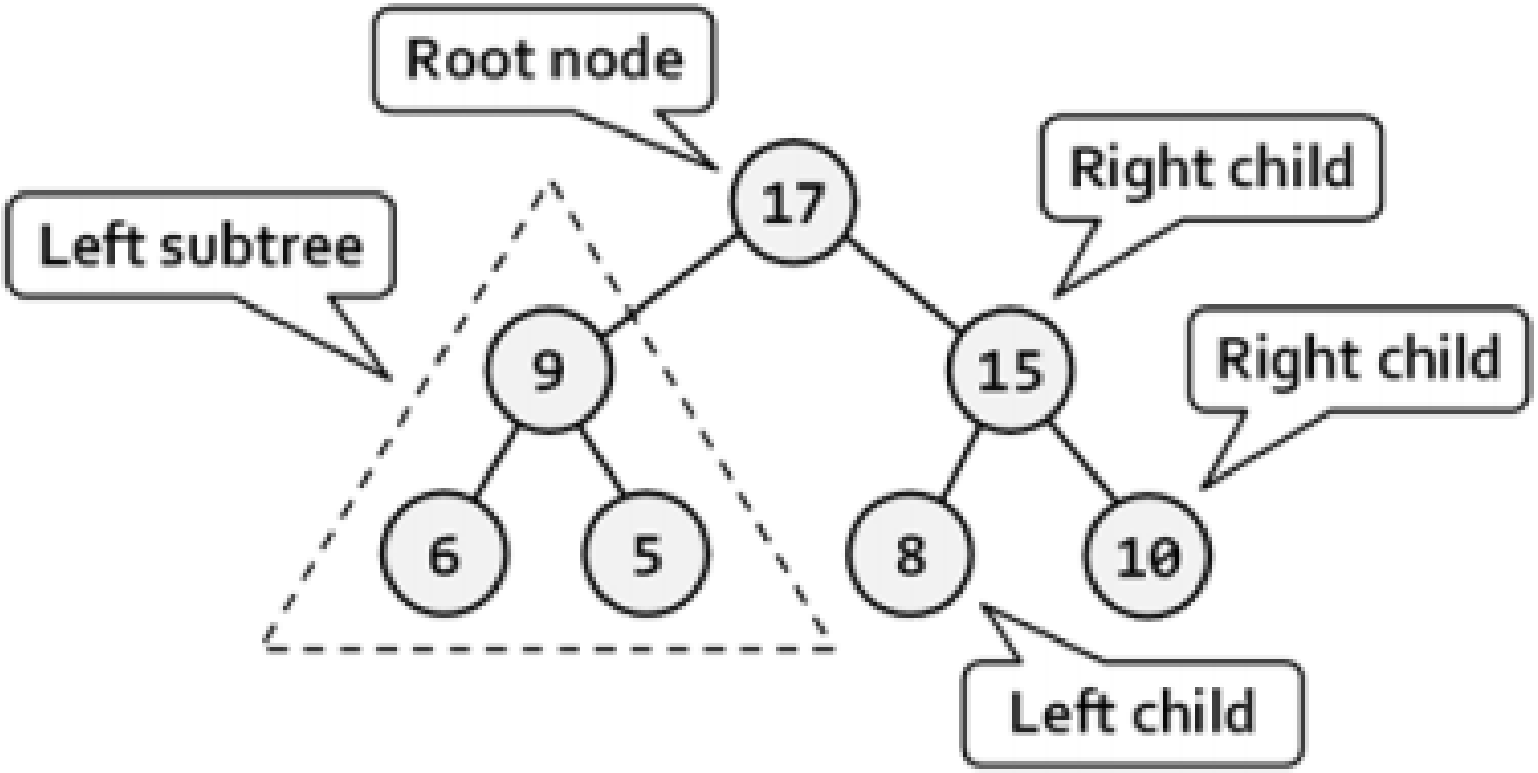
Link (YouTube): https://youtu.be/ozmE8G6YKww

Practice time: CodingBat



Try practicing writing recursive methods in Java using **CodingBat**, then practice writing them in IB pseudo code

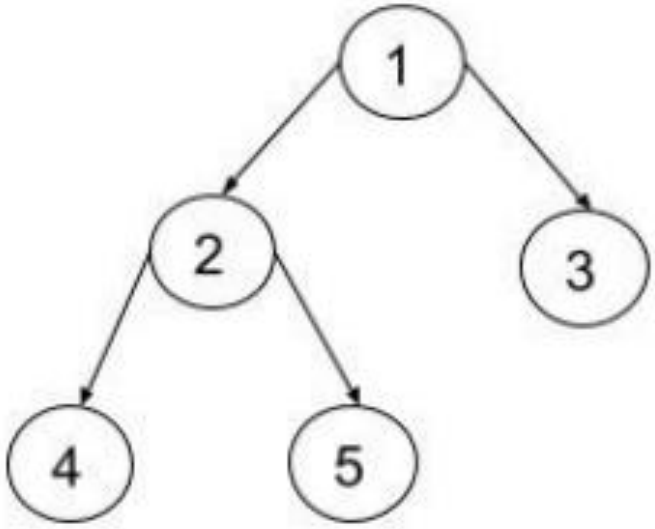
Recursion & Binary Trees



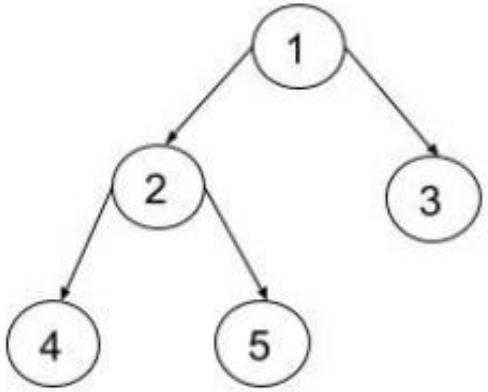
Exercise 1: Find max sum

Implement **`findMaxSum()`** method that find the maximum sum of all paths (each path has their own sum and find max sum of those sums). For example, the path **1** → **2** → **5** makes the max sum of 8 and 8 is the result.

```
int findMaxSum(Node n)
{
//insert code here
}
```



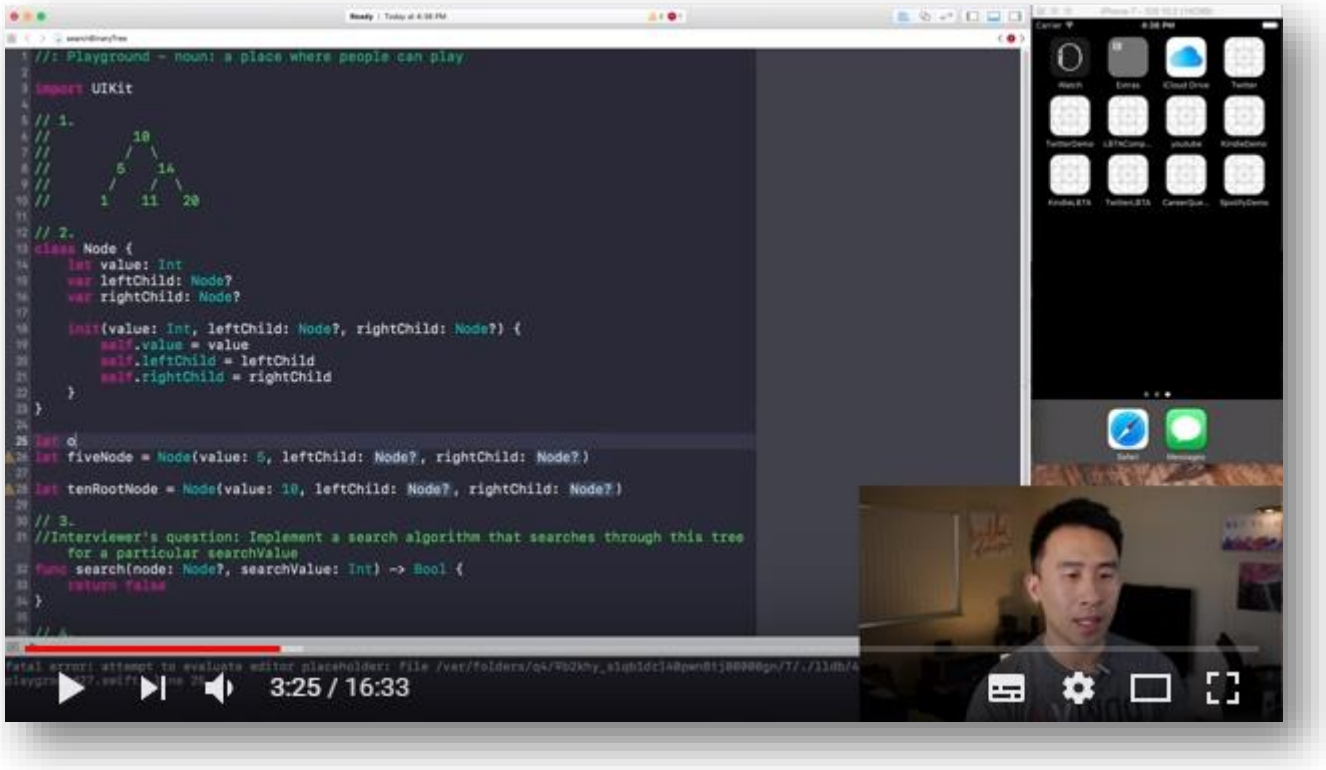
Exercise 1: Solution




```
int findMaxSum(Node n) {
    if (n == null) return 0;
    else {
        int sumleft = findMaxSum(n.left);
        int sumright = findMaxSum(n.right);
        if (sumleft > sumright)
            return n.data + sumleft;
        else
            return n.data + sumright;
    }
}
```

In this case, the path 1->2->5 makes sum of 8; 1->2>4 makes sum of 7; and 1->3 makes sum of 4. Therefore, 8 is the result.

Video: Recursively searching binary trees



Link (YouTube): https://youtu.be/beQEimO3-3E