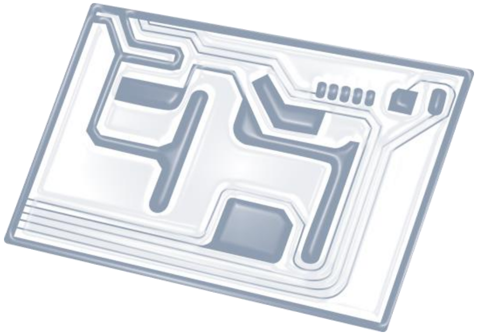
Abstract Data Structures

IB Computer Science

Content developed by
Dartford Grammar School
Computer Science Department

ORA ET LABORA
FOUNDED 1576

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

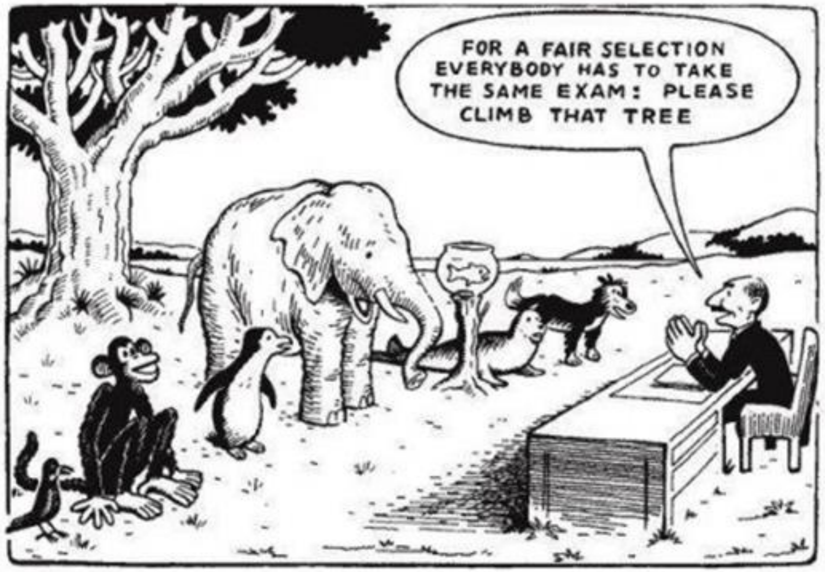
5.1.18 Define the term dynamic data structure

5.1.19 Compare the use of static and dynamic data structures

5.1.20 Suggest a suitable structure for a given situation

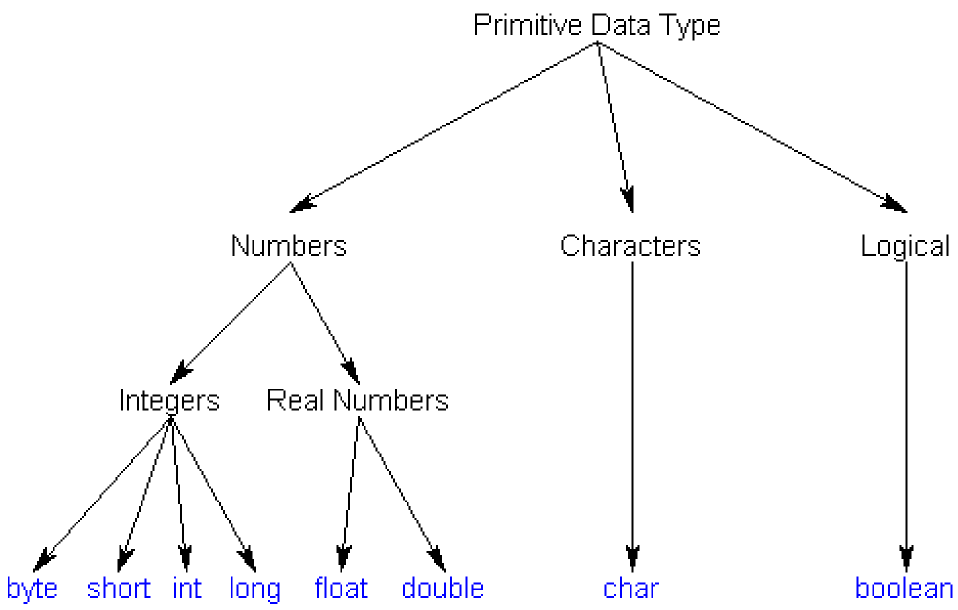
Topic 5.1.16

State the result of **inorder**, **postorder** and **preorder** tree traversal



Abstract Data Structures (ADTs)

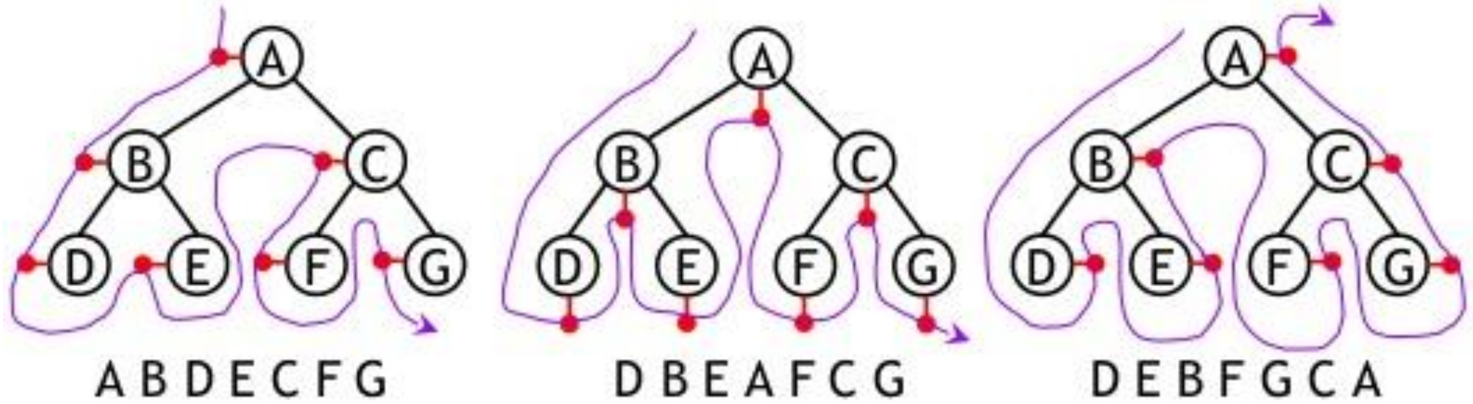
- 2D array
- Stack
- Queue
- Linked List
- **(Binary) Tree**
- Recursion



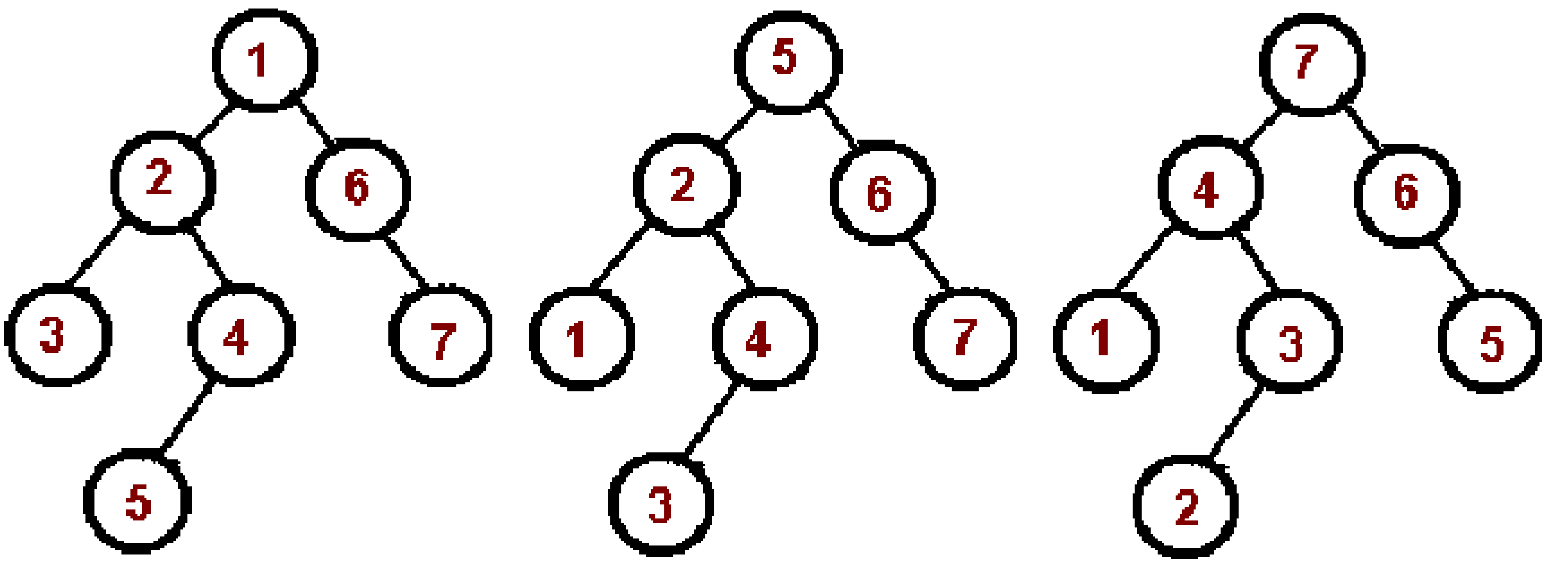
The "Flag" Rule

The order in which the nodes are visited during a tree traversal can be easily determined by imagining there is a "flag" attached to each node, as follows:

To traverse the tree, collect the flags:

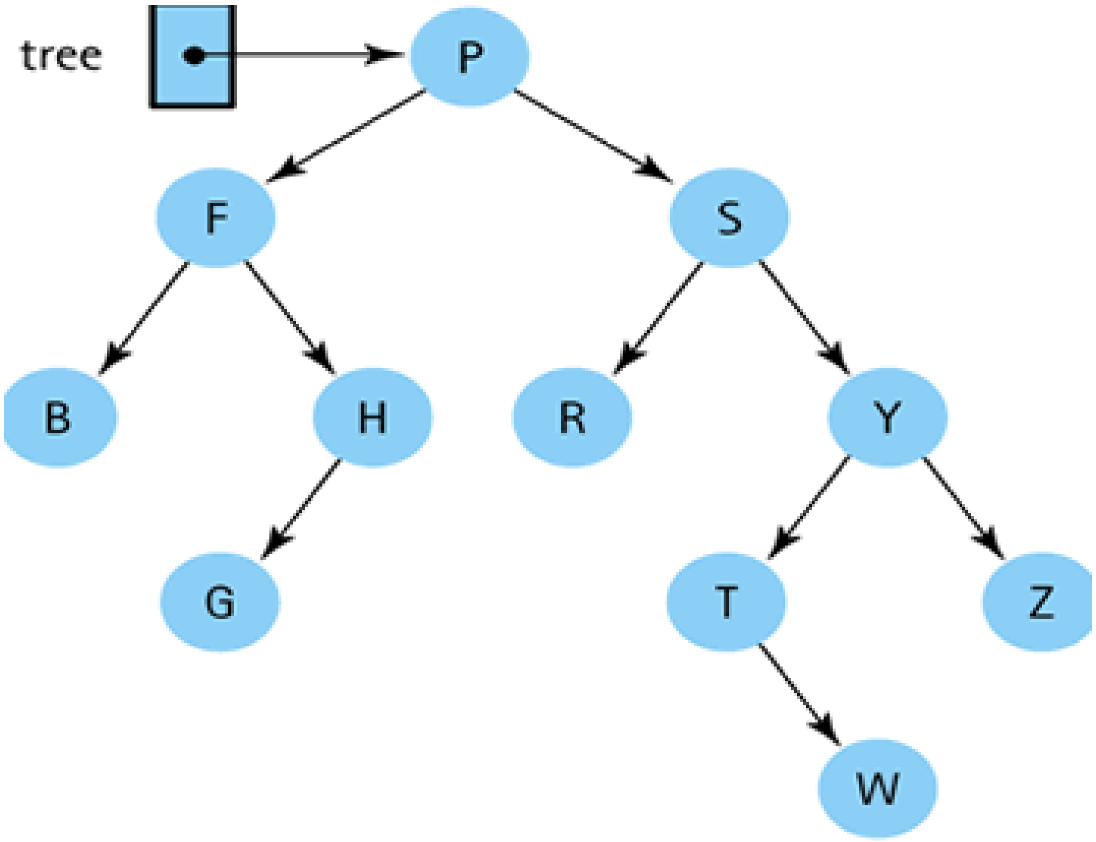


3 types of 'climbing' / traversal



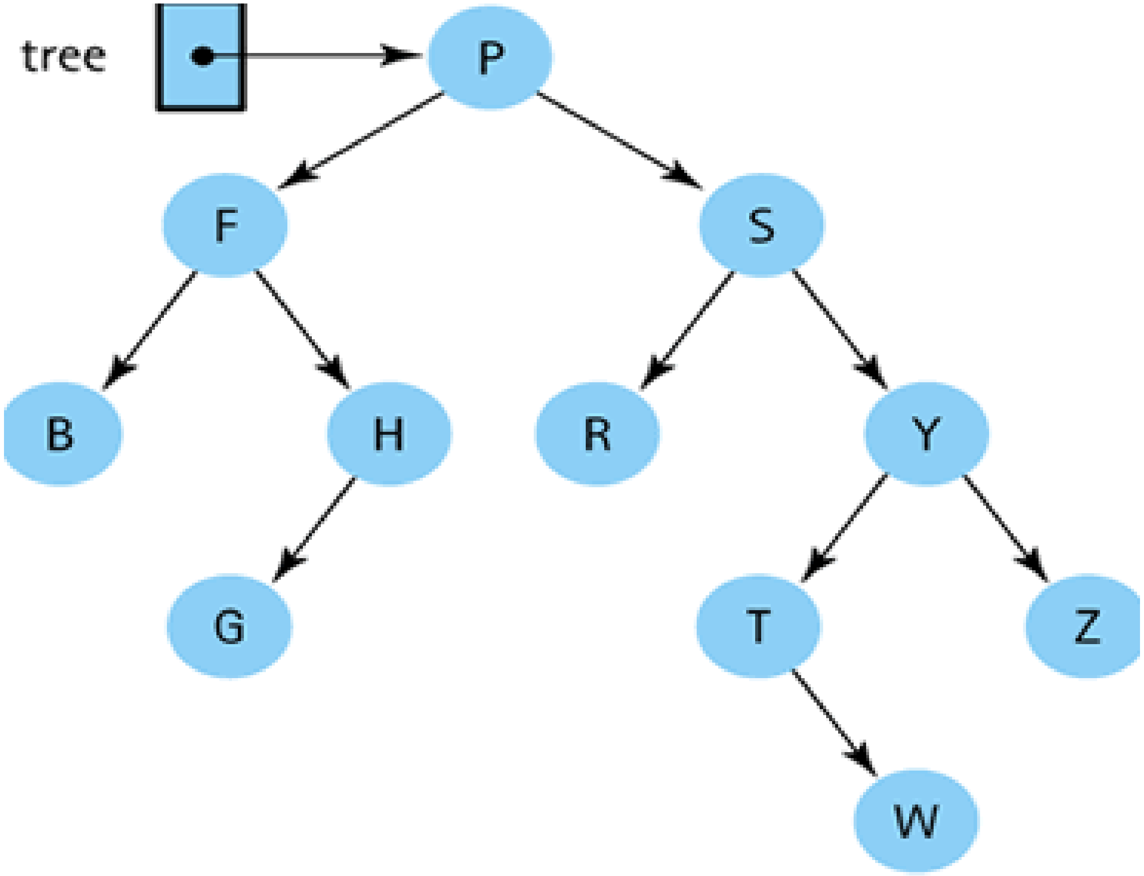
preorder inorder postorder

INORDER traversal



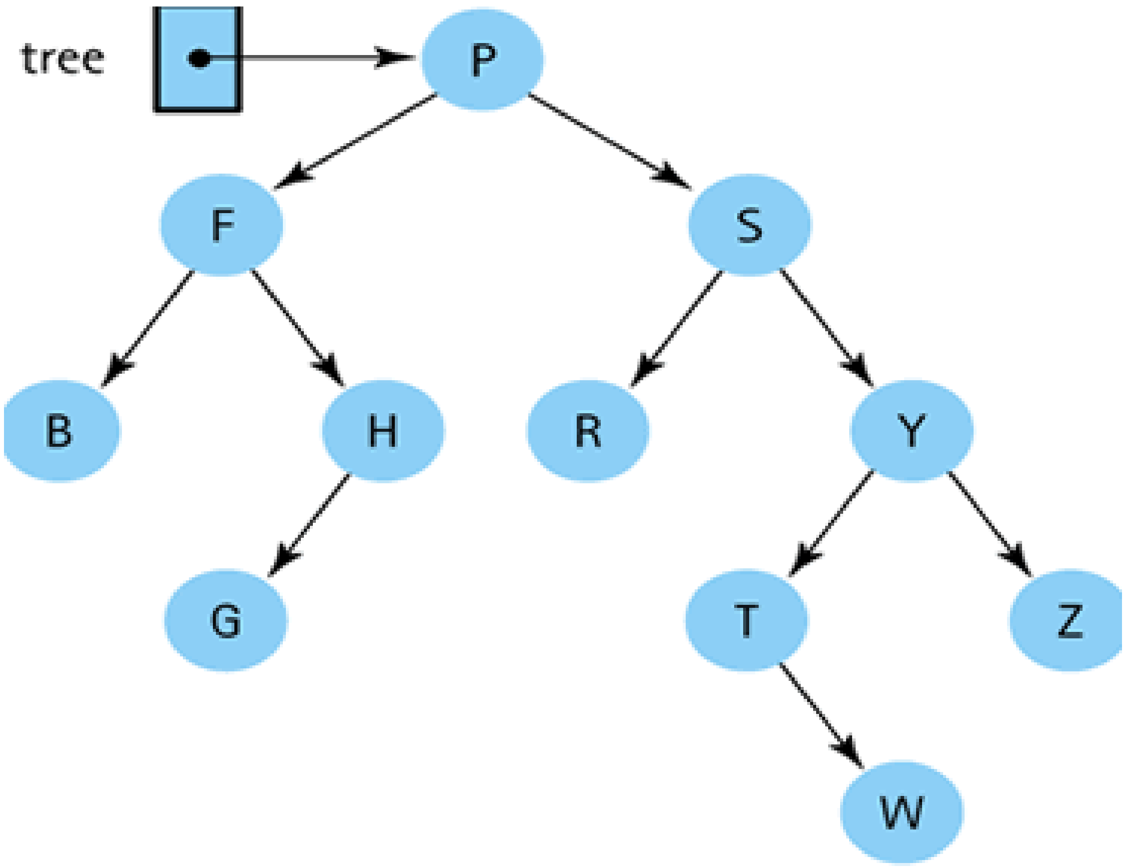
B F G H P R S T W Y Z

PREORDER traversal



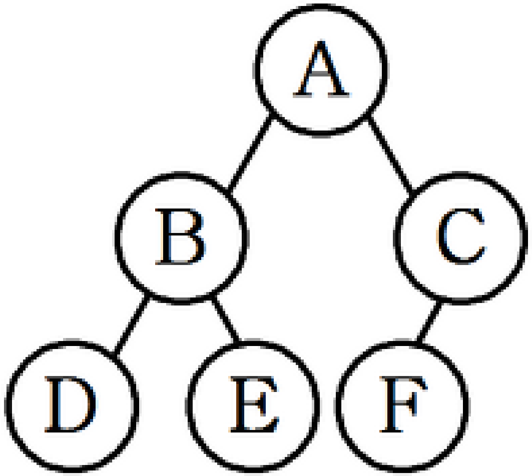
P F B H G S R Y T W Z

POSTORDER traversal



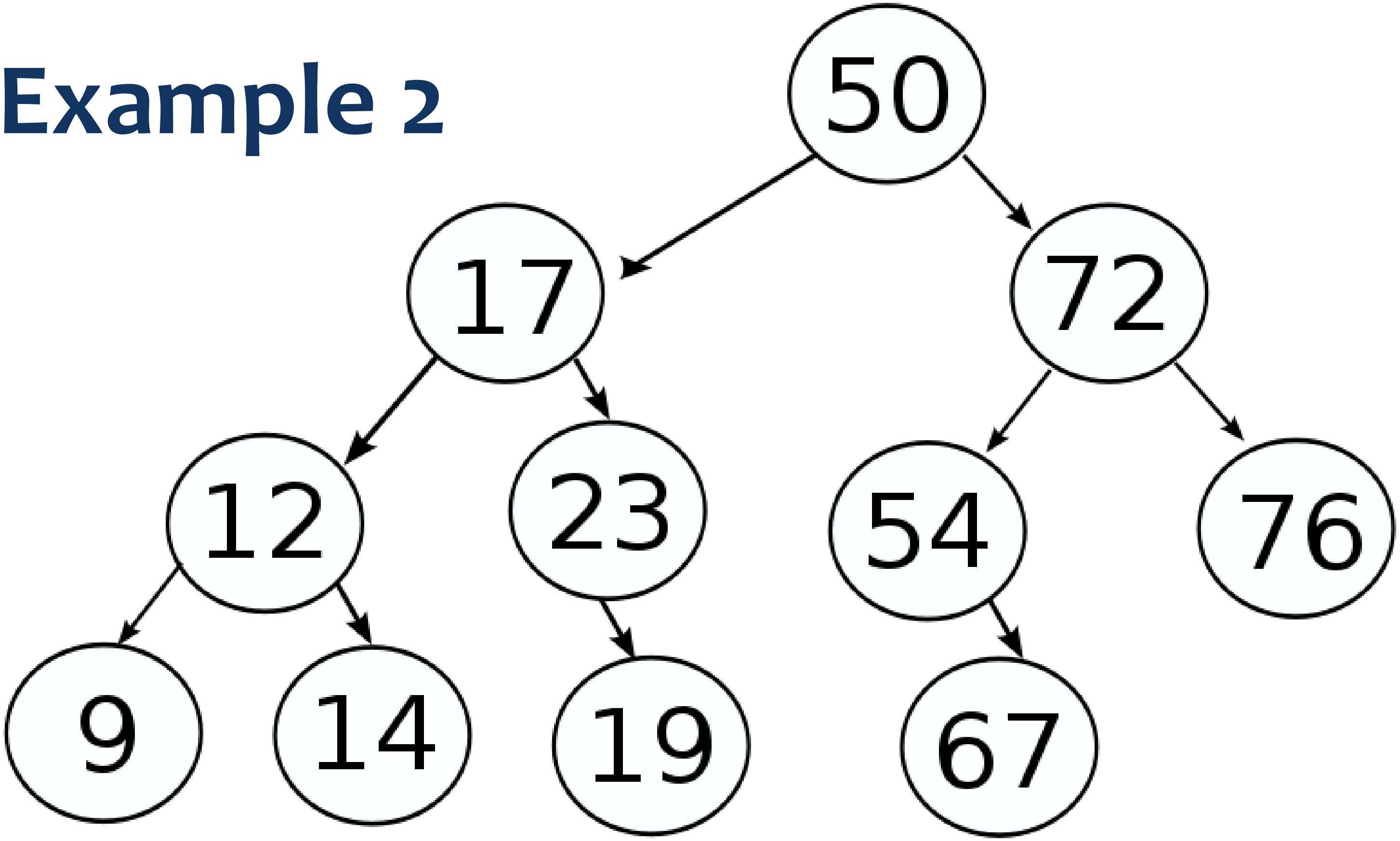
B G H F R W T Z Y S P

Example 1



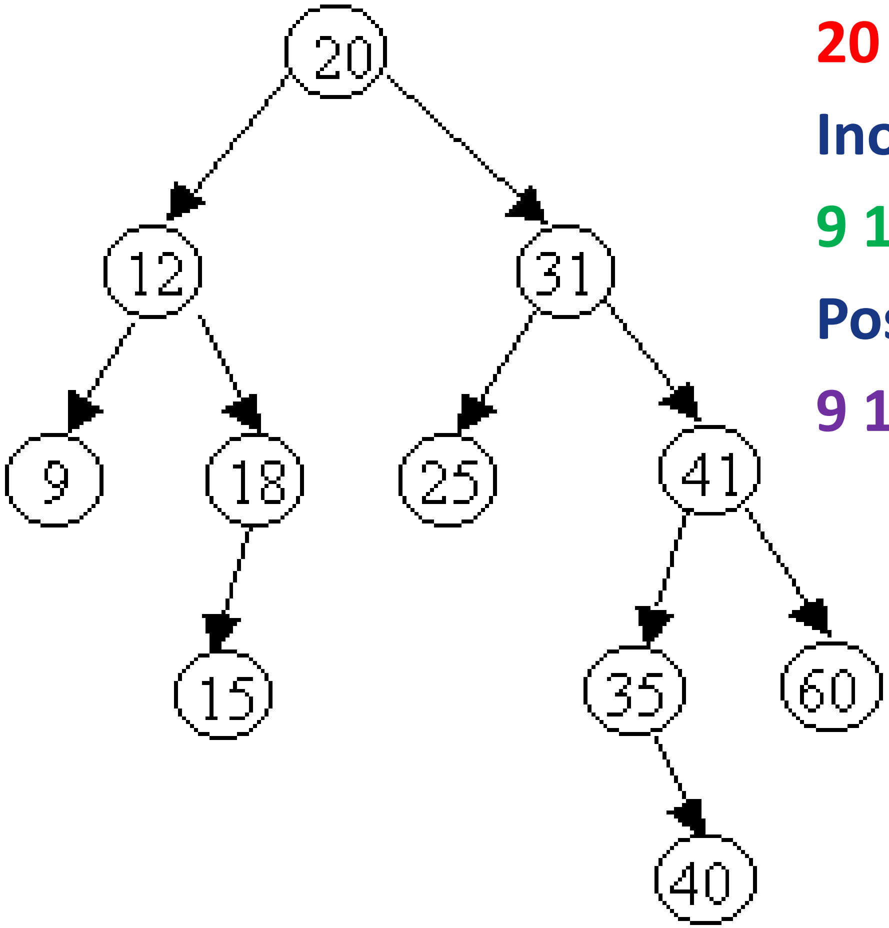
- Preorder (top start): **A B D E C F**
- Inorder (left start): **D B E A F C**
- Postorder (left start): **D E B F C A**

Example 2



- Preorder (top start): 50 17 12 9 14 23 19 72 54 67 76
- Inorder (left start): 9 12 14 17 19 23 50 54 67 72 76
- Postorder (left start): 9 14 12 19 23 17 67 54 76 72 50

Example 3



Preorder (top start):

20 12 9 18 15 31 25 41 35 40 60

Inorder (left start):

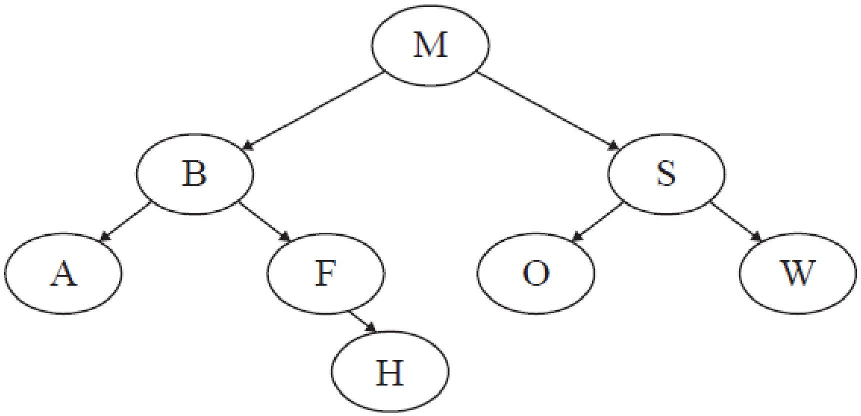
9 12 15 18 20 25 31 35 40 41 60

Postorder (left start):

9 15 18 12 25 40 35 60 41 31 20

Exam question, (1 mark each)

10. Consider the following binary search tree.



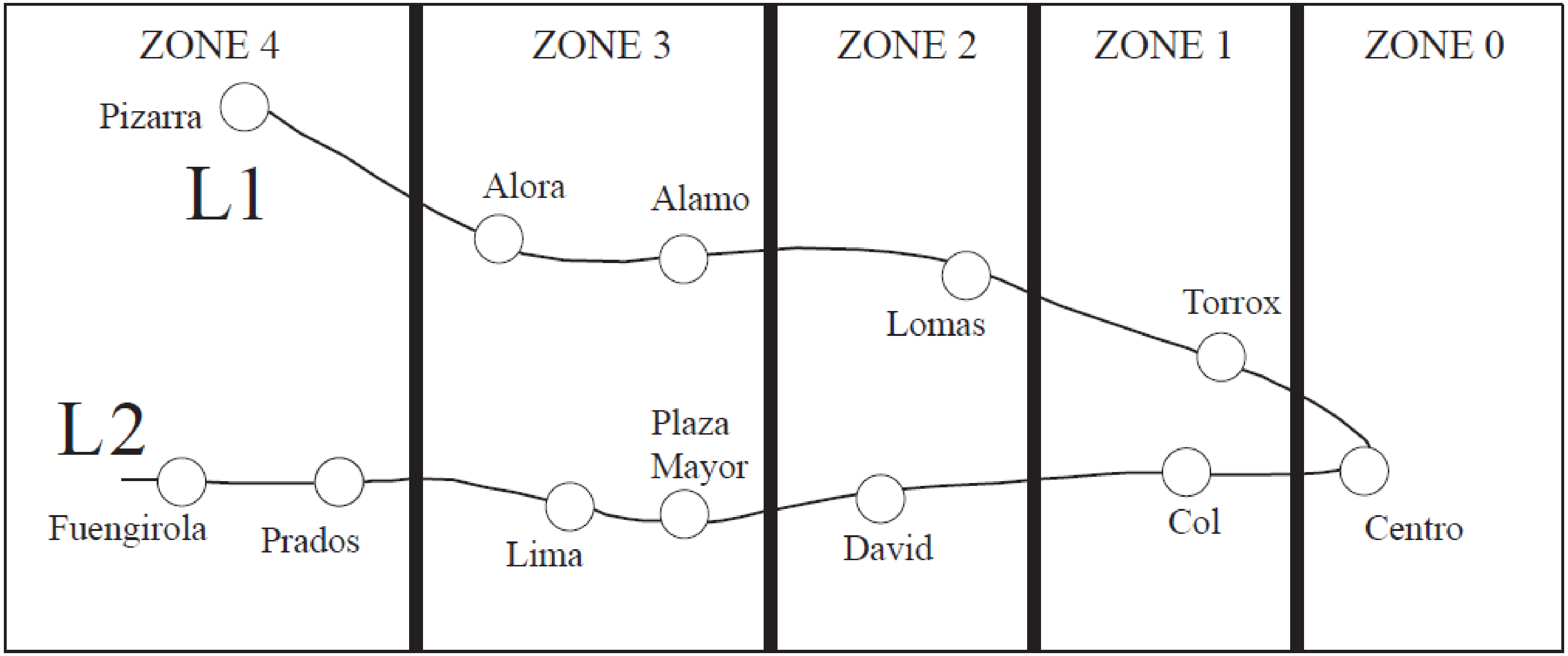
(a) State the order in which data will be listed using *preorder* traversal. *[1 mark]*

(b) State the number of leaf nodes in the tree. *[1 mark]*

(c) Construct the tree after adding the node L. *[1 mark]*

A blended question (paper 1)

A suburban railway system for a large city in Southern Europe consists of two lines **L1** and **L2**, which meet at the station Centro, where passengers can change from one line to the other. The system is shown below.



Question (a)

Each station is located in a particular zone, and the total number of zones in which the journey takes place determines the train fare. Note, if a passenger starts in **Zone 1**, goes to **Zone 0** and then back to **Zone 1**, the journey has taken place in **three** zones. Examples of the number of zones are shown below for different journeys.

Travelling from	Travelling to	Number of zones
Lima	Plaza Mayor	1
Alora	Plaza Mayor	7
Lomas	Col	4

(a) State the number of zones in which the journey takes place when travelling from Alora to Fuengirola. *[1]*

(a) 8;

Question (b)

The data for each station (station name, line, zone) is stored on the system's server in the collection `TRAIN_DATA`. There are 12 stations in total. The first part of the collection is shown below.

```
Centro, L1, 0, Alora, L1, 3, Torrox, L1, 1, Col, L2, 1, ...
```

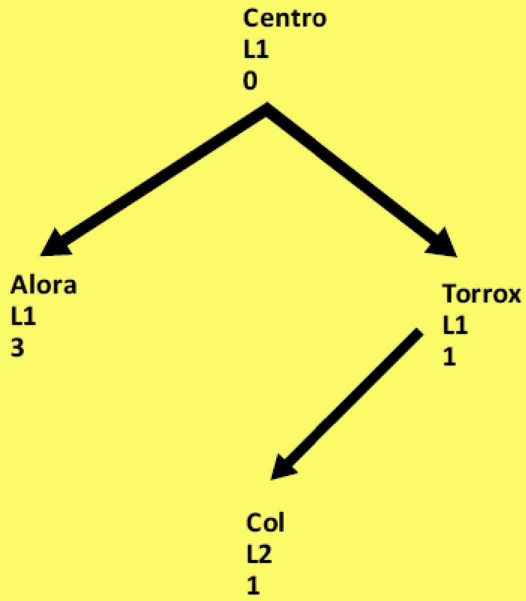
From this we can see that Alora is part of line L1 and is located in Zone 3.

At the start of each day, the data in `TRAIN_DATA` is read in to the binary tree `TREE`, in which each node will hold the data for one station. The binary tree will be used to search for a specific station's name.

(b) Sketch the binary tree after the station data from the first part of the collection, given above, has been added. *[3]*

Answer (b)

(b) *Centro* as root;
Station names in correct position;
All 3 items of data for each node;



[3 marks]

Question (c) – 1 mark

The `TRAIN_DATA` collection is also used to construct the one-dimensional array `STATIONS` (which only contains the list of station names sorted into alphabetical order), where `STATIONS[0] = Alamo`.

(c) State the value of `STATIONS[4]`. *[1]*

(c) David;

Question (d)

The two data structures (STATIONS and TREE) are now used to construct the two-dimensional array FARES containing the fares between stations, partly shown below. Note that the fare for travelling in **each** zone is €1.00.

FARES	Alamo	Alora	Centro	Col	...
Alamo	0	1.00	4.00	5.00	...
Alora	1.00	0	4.00	5.00	...
Centro	4.00	4.00	0	2.00	...
Col	5.00	5.00	2.00	0	...
...	...	...	...	...	... *etc*

(d) Calculate the fare for travelling from Torrox to Lima. *[1]*

(d) 5.00 (Euros);
Accept 5.

Question (e)

(e) Construct the algorithm that would calculate the fares for this two-dimensional array. You can make use of the following two sub-procedures:

```
• TREE.getZone(STATION)  // which returns the zone in which the
                         // station is located
• TREE.getLine(STATION)  // which returns the line on which the
                         // station is located
```

Your algorithm should make as few calculations as possible. *[9]*

Answer (e)

(e) *Award **[1 mark]** for each of the following 11 points, up to **[9 marks max]**.*

- *use of nested loops;*
- *use of nested loops with indices that avoid repeating calculations (as shown); (**Note:** outer loop can be to 11 if repeat calculations are avoided, with an IF statement)*
- *correct values retrieved from tree;*
- *check for same line;*
- *check if one of the stations is "Centro";*
- *check and change if negative/ use of absolute value;*
- *correct calculation for same line/one station is "Centro";*
- *correct calculation for different line;*
- *assign value to array;*
- *assign mirror value;*
- *assign value to diagonal;*

Answer (e) possible algorithm

```
loop N from 0 to 10
  STATION1 = STATION[N]
  AZ = TREE.getZone[STATION1]
  AL = TREE.getLine[STATION1]
  loop M from N+1 to 11  //start index changed so as not to repeat
                         //code
    STATION2 = STATION[M]
    BZ = TREE.getZone[STATION2]
    BL = TREE.getLine[STATION2]
    if AL = BL or STATION1 = "Centro" or STATION2 = "Centro"
       then              //on same line or passing through "Centro"
          X = AZ - BZ    //number of zones where the travel takes
                         //place can be negative
          if X<0 then      //allow use of absolute
            X = -X         //or equivalent,e.g. X = abs(AZ-BZ)
          endif
          X = X+1
        else  //on different lines
        X = AZ+BZ+1
    endif
    FARES[N][M]=X   //assigns value to 2D array
    FARES[M][N]=X   //assigns mirror value
  endloop
  FARES[N][N]=0     //leading diagonal
endloop
FARES[11][11]=0     //final entry
```

[9 marks]