1-Observe the following AVL balanced binary search tree:

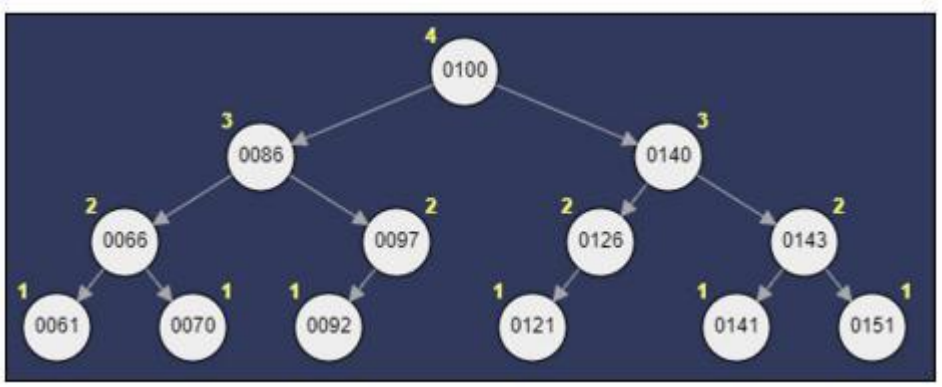
considering the insertion of the following elements (in order): 129, 134 and 136, analyze the following statements.

I.Causes a double rotation of the tree, right/left, which adds a new node to the second level of the tree.

II.It results in a simple rotation and increases the height of the tree.

III.After insertion, the computational complexity of the operations remains at O(log n), in the worst case, where n is the number of nodes in the tree.

It is correct what is stated in



a) II.

b) I and II.

c)I and III.

d) II and III.

two-Consider an AVL tree where the following tree is unbalanced after an insert operation:

a) Rotation to the right by 15, followed by a rotation to the left by 20.

b) Left rotation by 10, followed by a right rotation by 15.

c) Left rotation by 15, followed by a right rotation by 12.

d) Rotation to the right by 15, followed by a rotation to the left by 12.

e) It is not possible to balance the tree using rotations; it is necessary to rearrange the nodes in another way.

3-What is an AVL tree

a) a tree that is balanced and is a tree balanced in height

b) a tree that is unbalanced and is a tree balanced in height

c) a tree with three children

d) a tree with a maximum of 3 children

4-Consider a data structure*T*as an AVL binary tree. Characteristically, this data structure is a binary tree

a) balanced, in which, for any node of T, the heights of its two subtrees (left and right) differ by up to one unit.

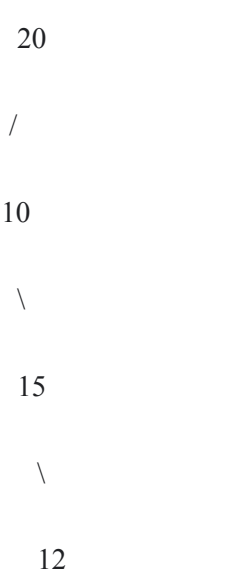
b) balanced, in which, for any node in T, the heights of its two subtrees (left and right) are always identical.

c) unbalanced, in which, for any node of T, the heights of its two subtrees (left and right) differ by up to one unit.

d) unbalanced, in which, for any node in T, the heights of its two subtrees (left and right) are always identical.

e) unbalanced, in which, for any node of T, the heights of its two subtrees (left and right) differ by exactly one unit.

5-Consider the following unbalanced AVL tree after an insert operation:



a) Double left-right rotation by 10, followed by a right rotation by 20.
b) Double right-left rotation by 15, followed by a left rotation by 20.
c) Double left-right rotation by 12, followed by a right rotation by 15.
d) Double right-left rotation by 10, followed by a left rotation by 12.
e) It is not possible to balance the tree using double rotations; it is necessary to rearrange the nodes in another way.

6-Height-balanced AVL tree means that, for each node in the tree, the difference between the heights of its sub-trees (right and left) will always be

a) less than or equal to 2.

b) equal to 0 or -1.

c) greater than 1.

d) equal to 1.

e) equal to -1, 0 or 1.

7-Given an empty AVL tree, how would you construct an AVL tree when a set of numbers is given without performing rotations?

a) simply build the tree with the given input

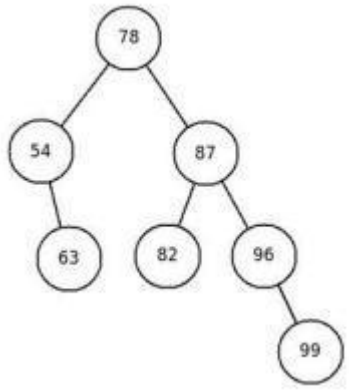
b) find the median of the set of given elements, make it the root and build the tree

c) use trial and error

d) use dynamic programming to build the tree

8-Consider the AVL tree below.

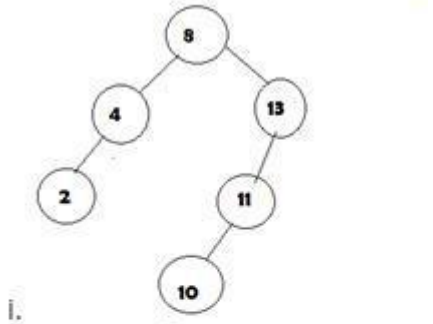
Mark the item that contains the route in pre-order after inserting a node containing the value 100.



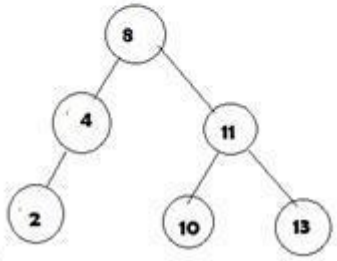
a) 54, 63, 78, 82, 87, 96, 99, 100
b) 63, 54, 78, 100, 99, 96, 82, 87
c) 63, 78, 54, 87, 82, 96, 99, 100
d) 78, 63, 54, 100, 99, 96, 87, 82
e) 78, 54, 63, 96, 87, 82, 99, 100

9-Which image(s) represent AVL tree

i.



ii.



a) just i

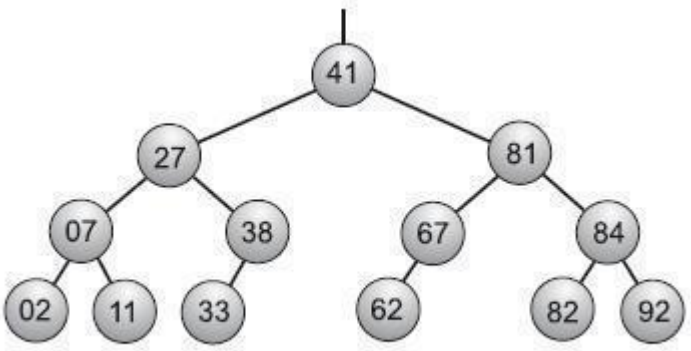
b)i and ii

c) only ii

d) i is not a binary search tree

10-Suppose the following AVL tree.

Inserting element 30 in this tree:



a) increases the depth of the tree after a rotation.

b) causes a rotation to the right.

c) leaves nodes 02 and 07 at the same level.

d) changes the root of the tree (node 41).

e) makes node 33 the parent of node 27.

11-Consider the left-left rotation pseudo code below, where the node contains value pointers to the left and right child nodes in addition to a height value, and the Height() function returns the height value stored in a specific node.

```
avltree leftrotation(avltreenode z):
  avltreenode w =x-left
  x-left=w-right
  w-right=x
  x-height=max(Height(x-left),Height(x-right))+1
  w-height=max(missing)+1
 return w
```

a) Height(w-left), height(x)
b) Height(w-right), height(x)
c) Height(w-left), x
d) Height(w-left)

12-Consider the following pseudo code and check whether it can check whether the binary search tree is of type AVL or not.

```
int avl(binarysearchtree root):
   if(not root)
     return 0
   left_tree_height = avl(left_of_root)

   if(left_tree_height== -1)
     return left_tree_height

   right_tree_height= avl(right_of_root)

   if(right_tree_height==-1)
     return right_tree_height
```

a) Yes, he can check it.
b) No, he can't check.
c) It is inconclusive
d) The code is incomplete

13-Select the alternativecorrectregarding the file structure.

a) A tree is considered unbalanced if, and only if, for any node, the height of its two sub-trees differs by at most one unit. Examples of balanced trees are AVL trees.

b) A tree is considered degenerate if, and only if, for any node, the height of its two sub-trees differs by at most one unit. Examples of balanced trees are AVL trees.

c) An AVL tree is a tree in which the heights of the left and right subtrees of each node differ by at least one unit.

d) When inserting into an AVL tree, a balancing process is used, which can be of 2 general types: Simple rotation or complex rotation.

e) A tree is considered balanced if, and only if, for any node, the height of its two sub-trees differs by at most one unit. Examples of balanced trees are AVL trees

14-An AVL tree is a data structure widely used to store data in memory. It has some properties that mean that its height has a very specific relationship with the number of elements stored in it. For a leaf, whose height is equal to one, we have an AVL tree with 6 nodes.

What is the maximum height this tree can be?

a) 6

b) 5

c) 4

d) 3

e) 2

15-What is the maximum height of an AVL tree with p nodes?

a) p

b) log(p)

c) log(p)/2

d) p/2

16-Regarding the Algorithm and data structure in the case of an AVL tree (or height-balanced tree), analyze the statements below, give values True (T) or False (F) and select the alternative that presents the correct sequence from top to bottom:

( ) An AVL tree is said to be balanced when, for each node in the tree, the difference between the heights of its sub-trees (right and left) is not greater than one.

( ) If the tree is not balanced, it must be balanced through single rotation or double rotation.

Select the correct alternative:

a) FF

b) FT

c) TF

d) TT

17 -In a balanced binary search tree of the AVL type, the heights of the two subtrees of any node differ by at most 1. The construction of a tree of this type, initially empty, through the successive insertion of nodes, uses a certain operation to maintain the desired balance when necessary. This operation is

a) Stacking.

b) unstacking.

c) concatenation.

d) rotation.

e) pruning.

18-Regarding data structures and trees, judge the next items.

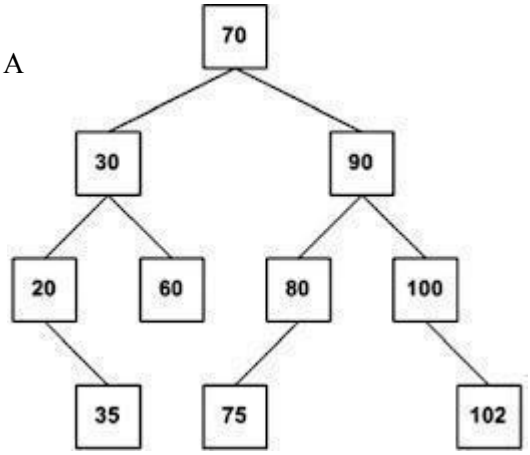
In an AVL tree (Adelson-Velsky and Landis), if the difference in height between the sub-trees of a node is equal to 2 and the difference in height between the child node of the unbalanced node is equal to -1, one must perform a double rotation with the child to the right and the parent to the left so that the tree is balanced again.

a) right

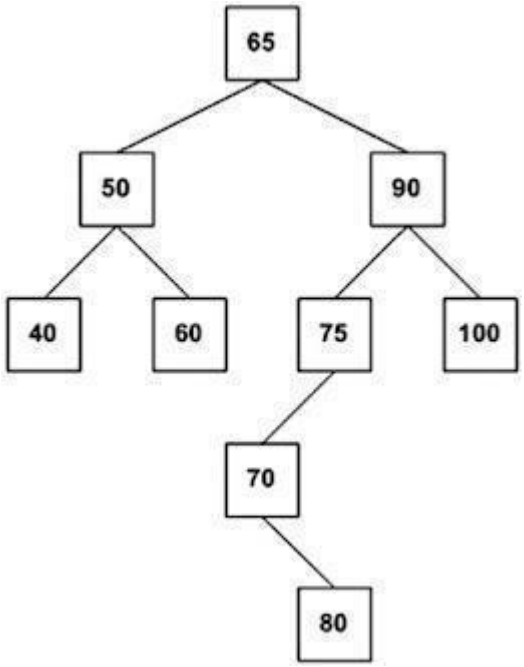
b) wrong

19 -Which figure represents an AVL tree?

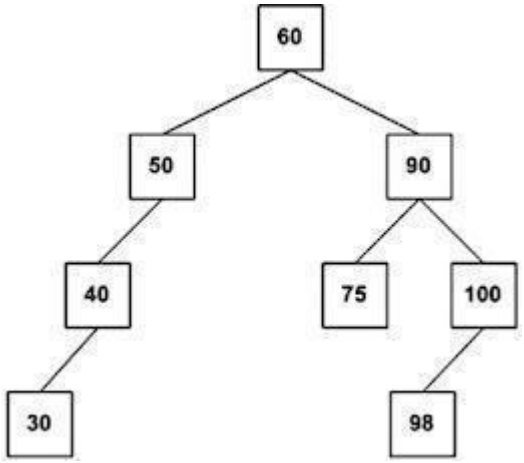
A



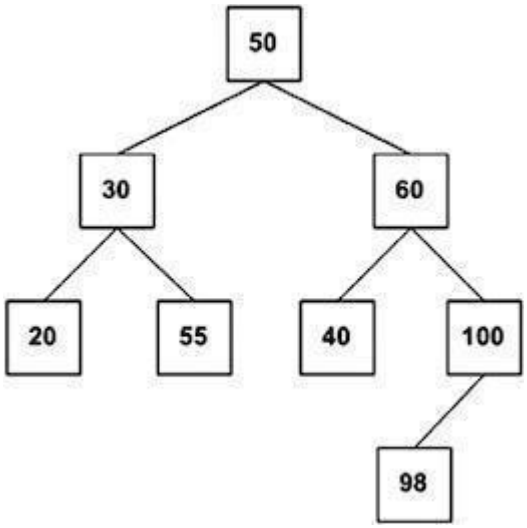
B



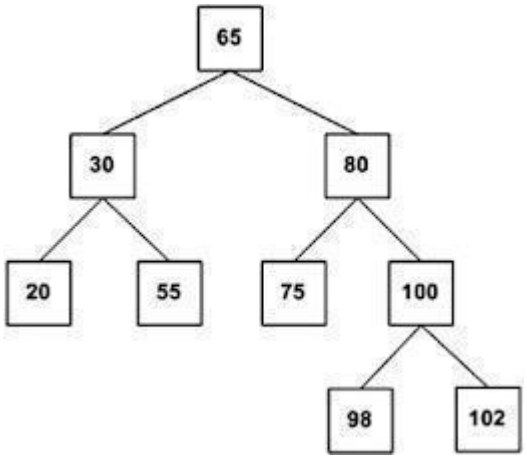
C



D



E



20 -Let T be an empty AVL (Adelson-Velski and Landis) balanced tree. Assuming that the elements 5, 10, 12, 8, 7, 11 and 13 are inserted in this order in T, the sequence that corresponds to a traversal of T in pre-order is

a) 10, 8, 5, 7, 12, 11 and 13.

b) 10, 7, 5, 8, 12, 11 and 13.

c) 5, 7, 8, 10, 11, 12 and 13.

d) 5, 8, 7, 11, 13, 12 and 10.

e) 5, 10, 12, 8, 7, 11 and 13.

21 -Why is it important to keep AVL trees balanced, and how are rotations used to maintain balance?

a) Balancing in AVL trees refers to keeping the tree always in a chaotic state, where the difference in height between the left and right subtrees of each node is arbitrarily large. This is crucial to ensure that search, insertion, and removal operations are unpredictable and challenging.

b) It is completely irrelevant to keep the AVL trees balanced; in fact, allowing the tree to degenerate into a completely unbalanced structure is highly desirable. This ensures that tree operations are a mysterious journey of unpredictable complexity.

c) Rotations are used to maintain balance in AVL trees. Single rotations (left or right) and double rotations (combinations of single rotations) are applied when insertions or deletions occur, adjusting the tree structure to preserve the balancing property

d) Rotations are useless and unnecessarily complicate AVL trees. Instead of adjusting the tree structure to preserve balance, it is preferable to apply random transformations to the nodes, making the tree an experiment in algorithmic chaos.

e) The balancing property in AVL trees is an algorithmic conspiracy. Completely ignoring this property and allowing the tree to grow wildly into unusual shapes is the true essence of efficiency in data structures.