1- // تعريف العقدة في الشجرة الثنائية  
class Node {  
 int data;  
 Node left;  
 Node right;  
  
 public Node(int data) {  
 this.data = data;  
 this.left = null;  
 this.right = null;  
 }  
}  
  
// تنفيذ الشجرة الثنائية  
class BinaryTree {  
 Node root;  
  
 public BinaryTree() {  
 this.root = null;  
 }  
  
 // إضافة عقدة جديدة إلى الشجرة  
 public void insert(int data) {  
 root = insertRecursive(root, data);  
 }  
  
 private Node insertRecursive(Node current, int data) {  
 if (current == null) {  
 return new Node(data);  
 }  
  
 if (data < current.data) {  
 current.left = insertRecursive(current.left, data);  
 } else if (data > current.data) {  
 current.right = insertRecursive(current.right, data);  
 }  
  
 return current;  
 }  
  
 // عرض الشجرة باستخدام ترتيب العرض الوسطي  
 public void inorderTraversal() {  
 inorderTraversalRecursive(root);  
 }  
  
 private void inorderTraversalRecursive(Node node) {  
 if (node != null) {  
 inorderTraversalRecursive(node.left);  
 System.*out*.print(node.data + " ");  
 inorderTraversalRecursive(node.right);  
 }  
 }  
}  
  
// استخدام الشجرة الثنائية  
public class Main {  
 public static void main(String[] args) {  
 BinaryTree tree = new BinaryTree();  
  
 // إدخال العقد في الشجرة  
 tree.insert(50);  
 tree.insert(30);  
 tree.insert(20);  
 tree.insert(40);  
 tree.insert(70);  
 tree.insert(60);  
 tree.insert(80);  
  
 // عرض الشجرة بترتيب العرض الوسطي  
 System.*out*.println("Inorder traversal:");  
 tree.inorderTraversal();  
 }

- a. صحيح. يجب أن يكون لدى الشجرة الثنائية على الأقل عقدة واحدة، مما يجعلها غير فارغة.

b. صحيح. يُعتبر مستوى العقدة الجذرية 0 في الشجرة الثنائية، ويتم عد المستويات اللاحقة من هذا الرقم.

c. صحيح. إذا كان لدى الشجرة عقدة واحدة فقط، فإنه يمكن اعتبارها تحتوي على 0 مستويات وبالتالي يكون ارتفاعها 0.

d. خاطئ. ترتيب العرض الوسطي (inorder traversal) للشجرة الثنائية لا ينتج دائمًا عن ترتيب البيانات بترتيب تصاعدي. في العرض الوسطي، يتم زيارة الجزء الأيسر من الشجرة أولاً، ثم العقدة الجذرية، ومن ثم الجزء الأيمن من الشجرة. الترتيب الناتج يعتمد على ترتيب العقد في الشجرة، وقد يكون مرتبًا تصاعديًا أو غير ذلك.

**شجرة ثنائية**

**الشكل 1:**

A

/ \

B C

/ \ / \

D E F G

**الأسئلة:**

1. ما هي العقدة LA في الشجرة الفرعية اليسرى للعقدة A؟
2. ما هي العقدة RA في الشجرة الفرعية اليمنى للعقدة A؟
3. ما هي العقدة RB في الشجرة الفرعية اليمنى للعقدة B؟
4. أدرج عقد الشجرة الثنائية هذه في تسلسل داخلي.
5. أدرج عقد الشجرة الثنائية هذه في تسلسل سابق.
6. أدرج عقد الشجرة الثنائية هذه في تسلسل لاحق.
7. ------------------------
8. **LA، العقدة في الشجرة الفرعية اليسرى لـ A هي D.**
9. **RA، العقدة في الشجرة الفرعية اليمنى لـ A هي B.** (لاحظ أنه يبدو أن هناك خطأ في السؤال، حيث أن B موجودة بالفعل في الشجرة الفرعية اليسرى لـ A، وليس في الشجرة الفرعية اليمنى.)
10. **RB، العقدة في الشجرة الفرعية اليمنى لـ B هي F.**
11. **تسلسل داخلي:** D، B، F، G، E، A، C
12. **تسلسل سابق:** A، B، D، F، G، E، C
13. **تسلسل لاحق:** D، F، G، B، E، C، A

**ملاحظة:**

* **الشجرة الفرعية اليسرى:** هي الشجرة التي تتفرع من العقدة اليسرى للعقدة الرئيسية.
* **الشجرة الفرعية اليمنى:** هي الشجرة التي تتفرع من العقدة اليمنى للعقدة الرئيسية.
* **التسلسل الداخلي:** هو تسلسل يتم فيه زيارة العقدة اليسرى أولاً، ثم العقدة الرئيسية، ثم العقدة اليمنى.
* **التسلسل السابق:** هو تسلسل يتم فيه زيارة العقدة الرئيسية أولاً، ثم العقدة اليسرى، ثم العقدة اليمنى.
* **التسلسل اللاحق:** هو تسلسل يتم فيه زيارة العقدة اليسرى أولاً، ثم العقدة اليمنى، ثم العقدة الرئيسية.

3-

**مسار من العقدة 80 إلى العقدة 79:**

مسار من العقدة ذات المعلومات 80 إلى العقدة ذات المعلومات 79 هو:

80 -> 70 -> 79

**2. إدراج العقدة 35:**

العقد التي تمت زيارتها بواسطة دالة insert لإدراج 35:

1. 80
2. 55
3. 52
4. 48 (يتم الإدراج هنا)

الشجرة المحدثة بعد إدراج 35:

50

30 80

25 40 55 98

48 52 35 85 110

45 70 90

65 79

75

**3. حذف العقدة 52:**

خلف العقدة 52 هو 58. بعد حذف 52 وترقية 58، تصبح الشجرة:

50

30 80

25 40 55 98

48 58 85 110

45 70 90

65 79

75

**4. حذف العقدة 40:**

خلف العقدة 40 هو 45. بعد حذف 40 وترقية 45، تصبح الشجرة:

50

30 80

25 45 55 98

48 58 85 110

70 90

65 79

75

**5. حذف العقد 80 و 58:**

**حذف 80:**

* يصبح الطفل الأيسر لـ 80 (70) الجذر الجديد.
* يصبح 79 الطفل الأيمن لـ 70.
* يصبح 98 الطفل الأيسر لـ 85.

الشجرة بعد حذف 80:

70

30 85

25 45 98 110

48 58 90

65 79

75

**حذف 58:**

* بما أن 58 ليس لديها أطفال، يتم إزالتها ببساطة.

الشجرة النهائية بعد حذف كل من 80 و 58:

70

30 85

25 45 98 110

48 90

65 79

75

**ملاحظات:**

* **خلف العقدة:** هو أصغر عقدة في الشجرة الفرعية اليمنى للعقدة.
* **ترقية العقدة:** هي عملية نقل العقدة إلى مكانة أعلى في الشجرة.

4-

public static int nodeCount(Node root) {

// حالة الأساس: الشجرة الفارغة تُرجع 0

if (root == null) {

return 0;

}

// عد العقد بشكل متكرر في الشجرتين الفرعيتين اليمنى واليسرى

int leftCount = nodeCount(root.left);

int rightCount = nodeCount(root.right);

// دمج العد وإضافة 1 للعقدة الحالية

return 1 + leftCount + rightCount;

}

* - **حالة الأساس:** إذا كانت العقدة الجذرية (root) فارغة (أي أن الشجرة فارغة) ، فإنها تُرجع 0.
* **الحالة المتكررة:**
  + يتم عد العقد في الشجرتين الفرعيتين اليمنى واليسرى بشكل متكرر باستخدام دالة nodeCount.
  + يتم دمج عدد العقد في الشجرتين الفرعيتين مع إضافة 1 للعقدة الجذرية الحالية.

5-

public static int leavesCount(Node root) {  
 // حالة الأساس: شجرة فارغة أو عقدة ورقة تحتوي على ورقة واحدة  
 if (root == null || (root.left == null && root.right == null)) {  
 return 1;  
 }  
  
 // عد الأوراق في الشجرتين الفرعيتين اليمنى واليسرى  
 int leftCount = leavesCount(root.left);  
 int rightCount = leavesCount(root.right);  
  
 // دمج العد  
 return leftCount + rightCount;  
 }

- **شرح الدالة:**

1. **حالة الأساس:**
   * إذا كانت root فارغة (شجرة فارغة) أو ورقة (ليس لديها أطفال) ، فإنها تُرجع 1.
2. **الحالة المتكررة:**
   * إذا كانت root تحتوي على كل من العقد الفرعية اليمنى واليسرى ، فإنها:
     + تُعدّ الأوراق في الشجرتين الفرعيتين اليمنى واليسرى بشكل متكرر.
     + دمج عدد الأوراق من الشجرتين الفرعيتين.

6-

public class BinarySearchTree {  
  
 public static class Node {  
 int key;  
 Node left;  
 Node right;  
  
 public Node(int key) {  
 this.key = key;  
 }  
 }  
  
 public static Node insert(Node root, int key) {  
 if (root == null) {  
 return new Node(key);  
 }  
  
 if (key < root.key) {  
 root.left = *insert*(root.left, key);  
 } else if (key > root.key) {  
 root.right = *insert*(root.right, key);  
 }  
  
 return root;  
 }  
  
 public static void printTree(Node root) {  
 if (root == null) {  
 return;  
 }  
  
 System.*out*.print(root.key + " ");  
 *printTree*(root.left);  
 *printTree*(root.right);  
 }  
  
 public static void main(String[] args) {  
 Node root = null;  
 int[] keys = {30, 40, 24, 58, 48, 26, 11, 13};  
  
 for (int key : keys) {  
 root = *insert*(root, key);  
 System.*out*.println("Tree after inserting " + key + ":");  
 *printTree*(root);  
 System.*out*.println();  
 }  
 }  
}

- Tree after inserting 30:  
 30  
  
 Tree after inserting 40:  
 30 40  
  
 Tree after inserting 24:  
 30 24 40  
  
 Tree after inserting 58:  
 30 24 40 58  
  
 Tree after inserting 48:  
 30 24 40 48 58  
  
 Tree after inserting 26:  
 30 24 26 40 48 58  
  
 Tree after inserting 11:  
 30 24 11 26 40 48 58  
  
 Tree after inserting 13:  
 30 24 11 13 26 40 48 58