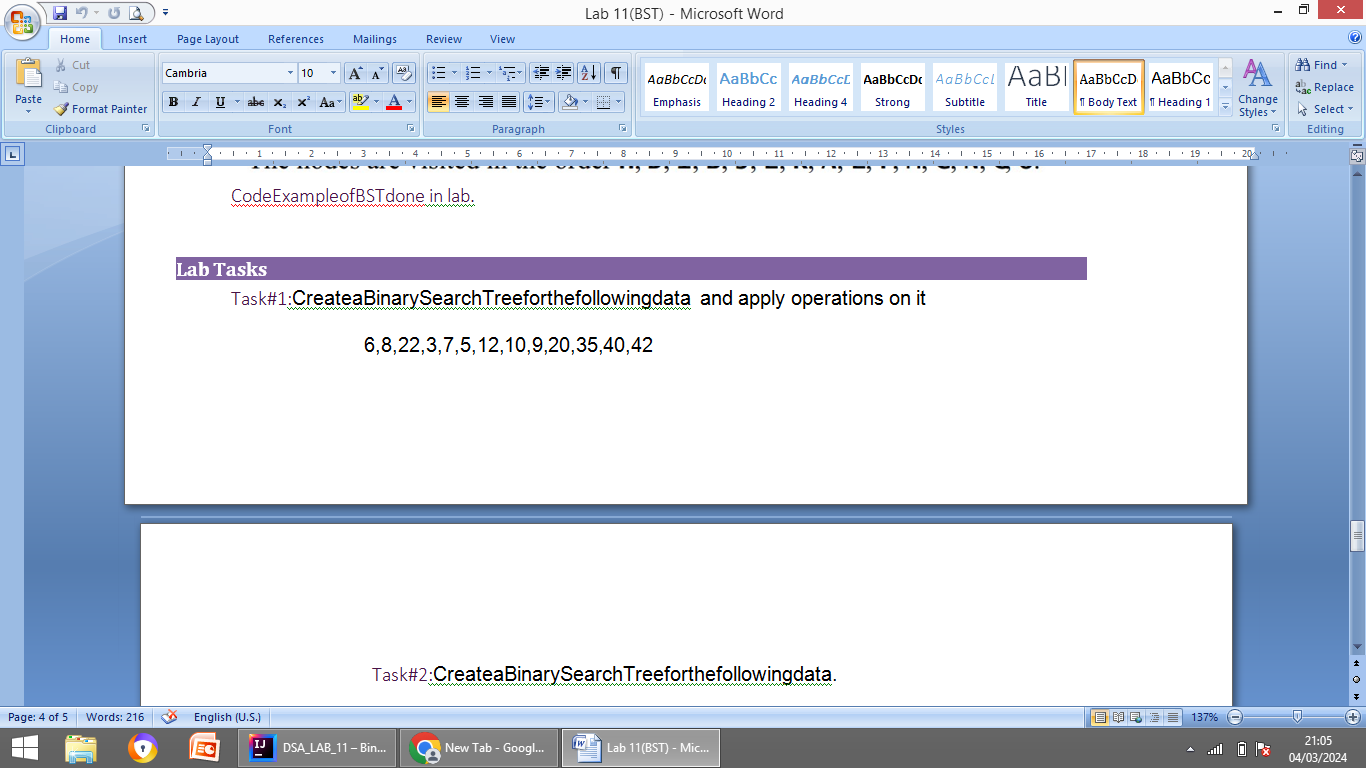
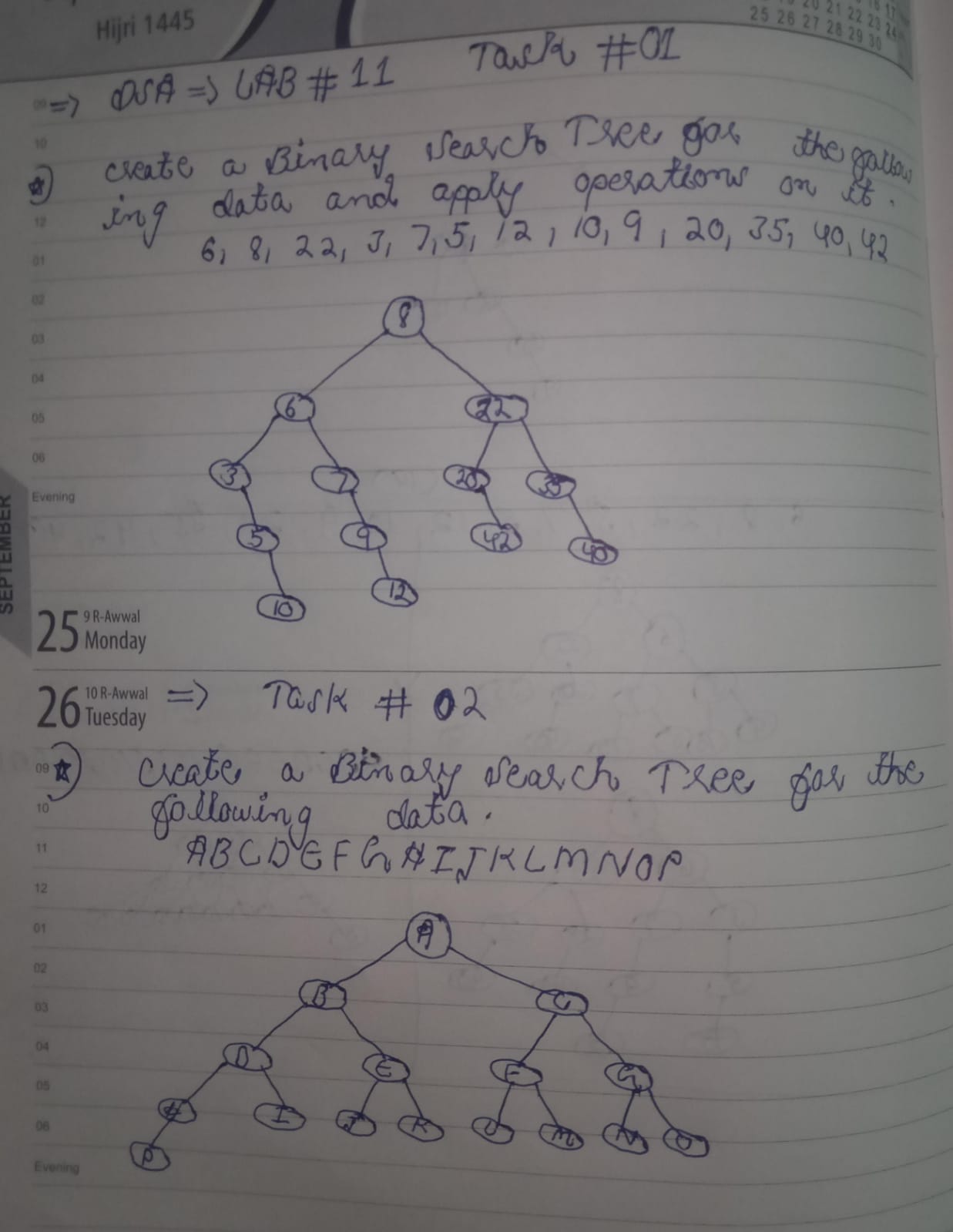
|  |  |
| --- | --- |
| ***Roll No*** | ***22SW040*** |
| ***Name*** | ***Farooque Sajjad*** |
| ***Section*** | ***01*** |
| ***Lab No*** | ***11(Trees)*** |

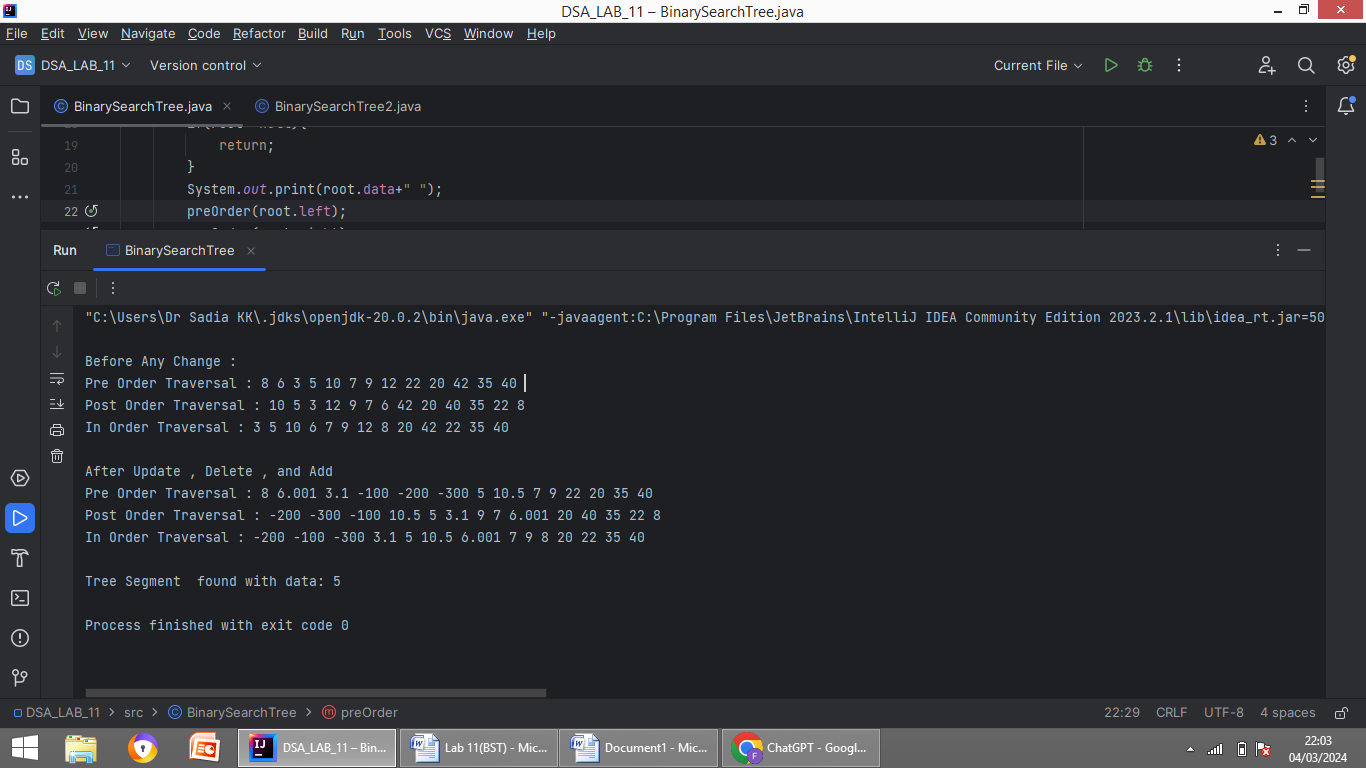


******

***Task 01***

// Lab No : 11 Task No 01  
  
public class BinarySearchTree {  
  
 BinarySearchTree left , right ;  
 Object data;  
  
 public BinarySearchTree(Object data) {  
 this.data = data;  
 }  
  
  
 public void preOrder(BinarySearchTree root){  
  
 if(root==null){  
 return;  
 }  
 System.*out*.print(root.data+" ");  
 preOrder(root.left);  
 preOrder(root.right);  
  
 }  
  
 public void postOrder(BinarySearchTree root){  
  
 if(root==null){  
 return;  
 }  
  
 postOrder(root.left);  
 postOrder(root.right);  
 System.*out*.print(root.data+" ");  
  
 }  
 public void InOrder(BinarySearchTree root){  
  
 if(root==null){  
 return;  
 }  
 InOrder(root.left);  
 System.*out*.print(root.data+" ");  
 InOrder(root.right);  
  
 }  
  
  
 public void add(Object newData) {  
 if (newData == null) {  
 return;  
 }  
  
 if (this.left == null) {  
 this.left = new BinarySearchTree(newData);  
 } else if (this.right == null) {  
 this.right = new BinarySearchTree(newData);  
 } else {  
  
 this.left.add(newData);  
 }  
 }  
  
  
 public boolean delete(Object target) {  
  
 BinarySearchTree targetNode = find(target);  
 if (targetNode == null) {  
 return false; }  
  
  
 targetNode.left = null;  
 targetNode.right = null;  
  
 return true;  
 }  
  
  
 public boolean update(Object target, Object newData) {  
 BinarySearchTree targetNode = find(target);  
 if (targetNode == null) {  
 return false;  
 }  
  
  
 targetNode.data = newData;  
 return true;  
 }  
  
  
  
  
 public BinarySearchTree find(Object target) {  
 if (data.equals(target)) {  
 return this;  
 }  
 if (left != null) {  
 BinarySearchTree leftResult = left.find(target);  
 if (leftResult != null) {  
 return leftResult;  
 }  
 }  
  
 if (right != null) {  
 BinarySearchTree rightResult = right.find(target);  
 if (rightResult != null) {  
 return rightResult;  
 }  
 }  
  
 return null;  
 }  
  
  
  
  
 public static void main(String[] args) {  
  
 BinarySearchTree t1 = new BinarySearchTree(8);  
 t1.left = new BinarySearchTree(6);  
 t1.right = new BinarySearchTree(22);  
 t1.left.left = new BinarySearchTree(3);  
 t1.left.right = new BinarySearchTree(7);  
 t1.right.left = new BinarySearchTree(20);  
 t1.right.right = new BinarySearchTree(35);  
 t1.left.left.right = new BinarySearchTree(5);  
 t1.left.left.right.right = new BinarySearchTree(10);  
 t1.left.right.right = new BinarySearchTree(9);  
 t1.left.right.right.right = new BinarySearchTree(12);  
 t1.right.left.right = new BinarySearchTree(42);  
 t1.right.right.right = new BinarySearchTree(40);  
  
 System.*out*.println();  
  
 System.*out*.println("Before Any Change : ");  
 System.*out*.print("Pre Order Traversal : ") ;  
 t1.preOrder(t1);  
 System.*out*.println();  
 System.*out*.print("Post Order Traversal : ") ;  
 t1.postOrder(t1);  
 System.*out*.println();  
 System.*out*.print("In Order Traversal : ") ;  
  
 t1.InOrder(t1);  
 System.*out*.println();  
 System.*out*.println();  
  
 System.*out*.println("After Update , Delete , and Add ");  
 t1.add(-100);  
 t1.add(-200);  
 t1.add(-300);  
  
 t1.delete(40);  
 t1.delete(20);  
 t1.delete(9);  
  
 t1.update(3,3.1);  
 t1.update(10,10.5);  
 t1.update(6,6.001);  
  
  
 System.*out*.print("Pre Order Traversal : ") ;  
 t1.preOrder(t1);  
 System.*out*.println();  
 System.*out*.print("Post Order Traversal : ") ;  
 t1.postOrder(t1);  
 System.*out*.println();  
 System.*out*.print("In Order Traversal : ") ;  
  
 t1.InOrder(t1);  
  
  
 System.*out*.println();  
 System.*out*.println();  
 BinarySearchTree result = t1.find(5);  
 if (result != null) {  
 System.*out*.println("Tree Segment found with data: " + result.data);  
 } else {  
 System.*out*.println("Tree Segment not found.");  
 }  
  
  
  
 }  
 }

***Output***



***Task 02***

// Lab No : 11 Task No 02  
  
public class BinarySearchTree2 {  
  
 BinarySearchTree2 left , right ;  
 Object data;  
  
 public BinarySearchTree2(Object data) {  
 this.data = data;  
 }  
  
  
 public void preOrder(BinarySearchTree2 root){  
  
 if(root==null){  
 return;  
 }  
 System.*out*.print(root.data+" ");  
 preOrder(root.left);  
 preOrder(root.right);  
  
 }  
  
 public void postOrder(BinarySearchTree2 root){  
  
 if(root==null){  
 return;  
 }  
  
 postOrder(root.left);  
 postOrder(root.right);  
 System.*out*.print(root.data+" ");  
  
 }  
 public void InOrder(BinarySearchTree2 root){  
  
 if(root==null){  
 return;  
 }  
 InOrder(root.left);  
 System.*out*.print(root.data+" ");  
 InOrder(root.right);  
  
 }  
  
  
 public static void main(String[] args) {  
 BinarySearchTree2 t1 = new BinarySearchTree2('A');  
 t1.left = new BinarySearchTree2('B');  
 t1.right = new BinarySearchTree2('C');  
 t1.left.left = new BinarySearchTree2('D');  
 t1.left.right = new BinarySearchTree2('E');  
 t1.right.left = new BinarySearchTree2('F');  
 t1.right.right = new BinarySearchTree2('G');  
 t1.left.left.left = new BinarySearchTree2('H');  
 t1.left.left.right = new BinarySearchTree2('I');  
 t1.left.left.left.left = new BinarySearchTree2('P');  
 t1.left.right.left = new BinarySearchTree2('J');  
 t1.left.right.right = new BinarySearchTree2('K');  
 t1.right.left.left = new BinarySearchTree2('L');  
 t1.right.left.right = new BinarySearchTree2('M');  
 t1.right.right.left = new BinarySearchTree2('N');  
 t1.right.right.right = new BinarySearchTree2('O');  
 System.*out*.println();  
  
  
 System.*out*.print("Pre Order Traversal : ") ;  
 t1.preOrder(t1);  
 System.*out*.println();  
 System.*out*.print("Post Order Traversal : ") ;  
 t1.postOrder(t1);  
 System.*out*.println();  
 System.*out*.print("In Order Traversal : ") ;  
  
 t1.InOrder(t1);  
 }  
  
  
}

***Output***

