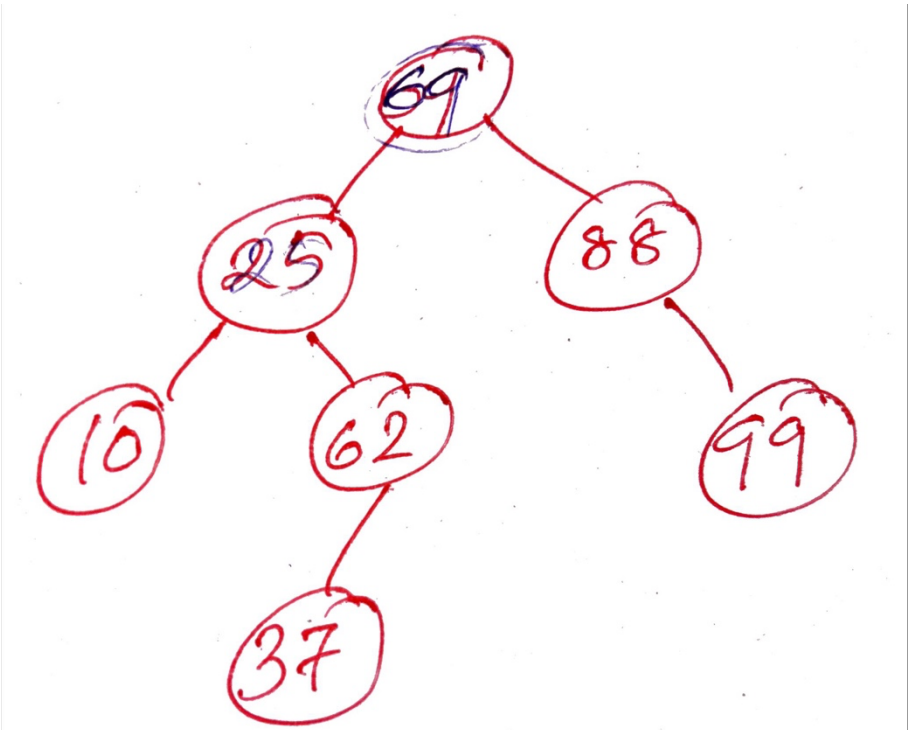
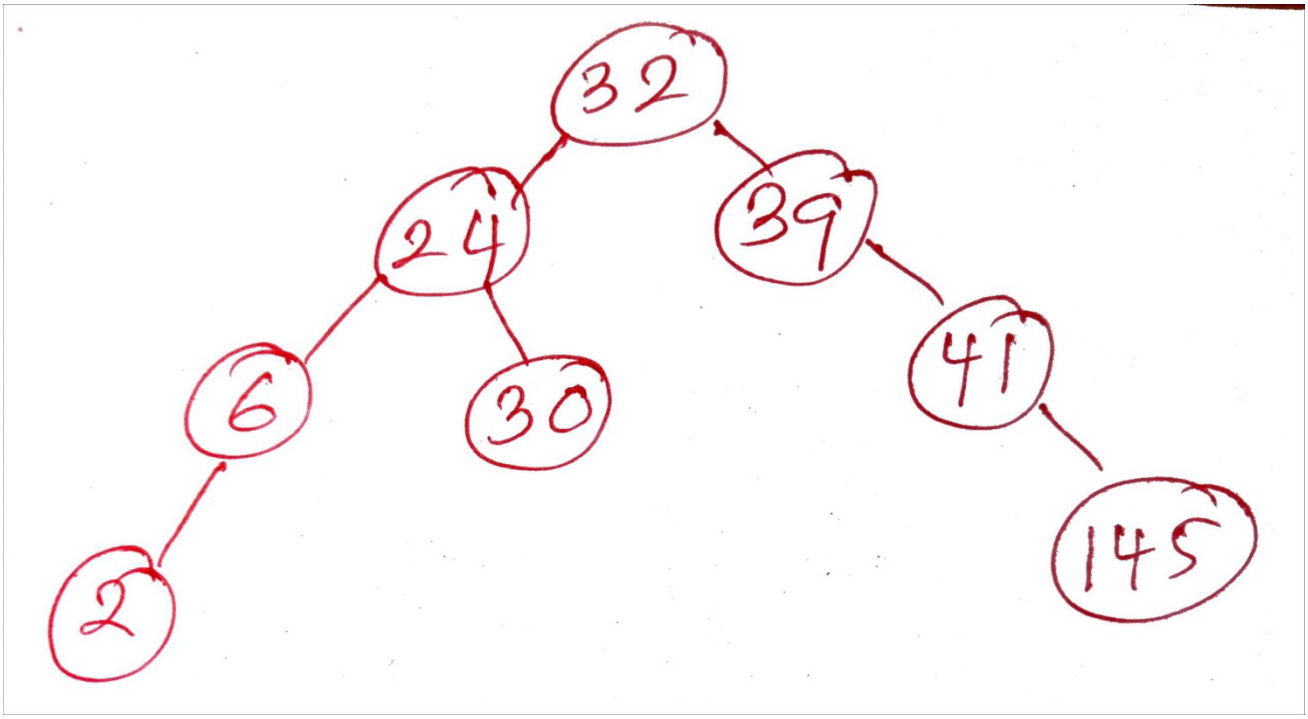
Question 01.

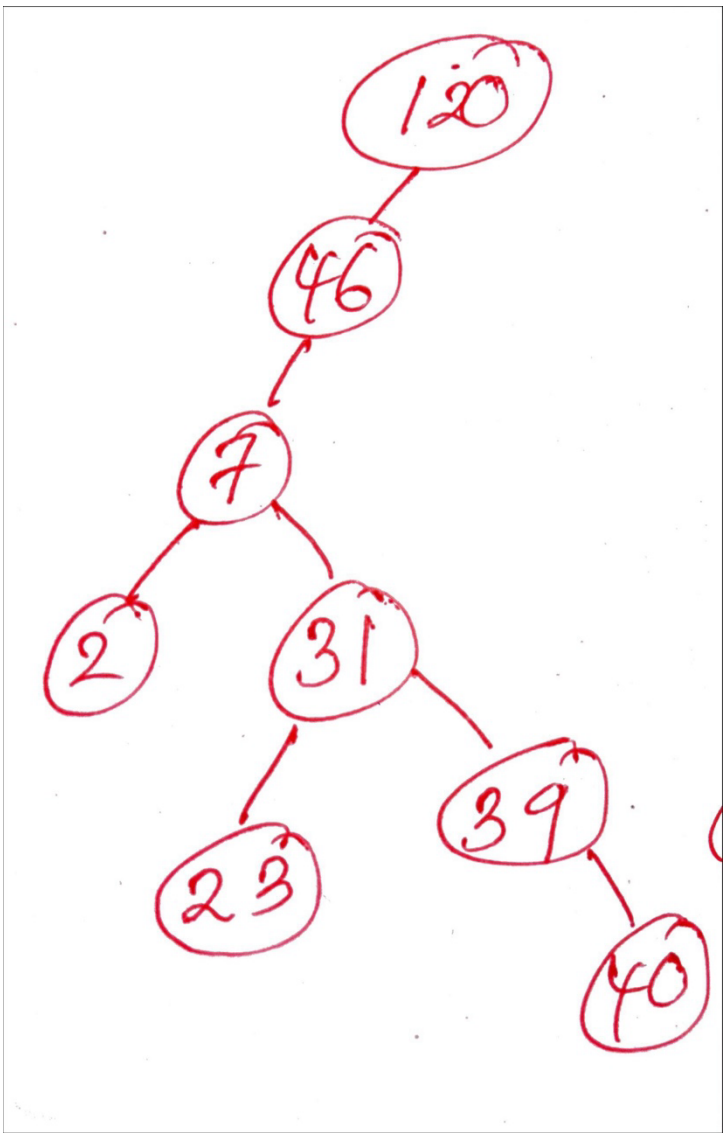
i. 69,25,62,88,10,37,99



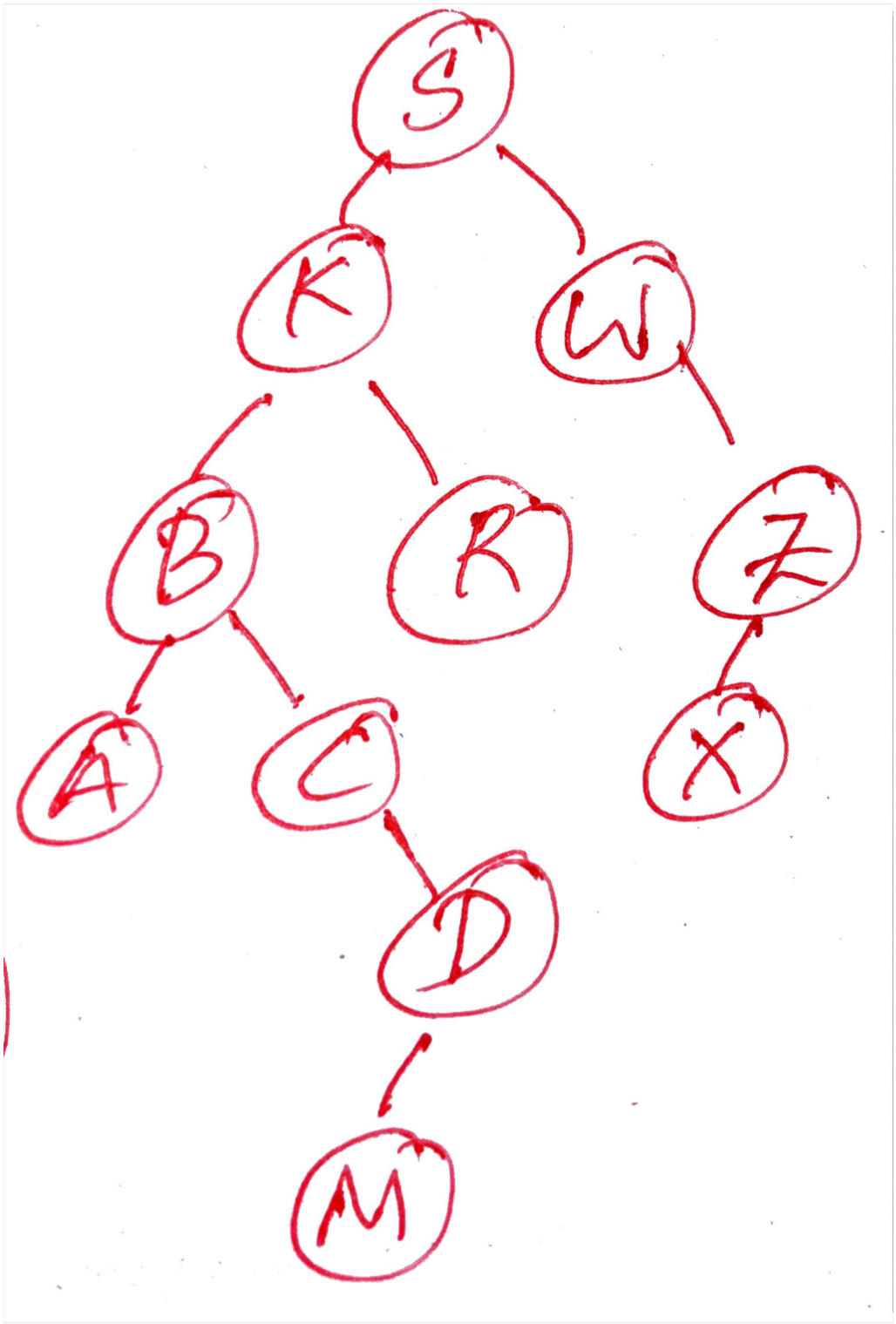
ii. 32,24,6,2,39,41,30,145



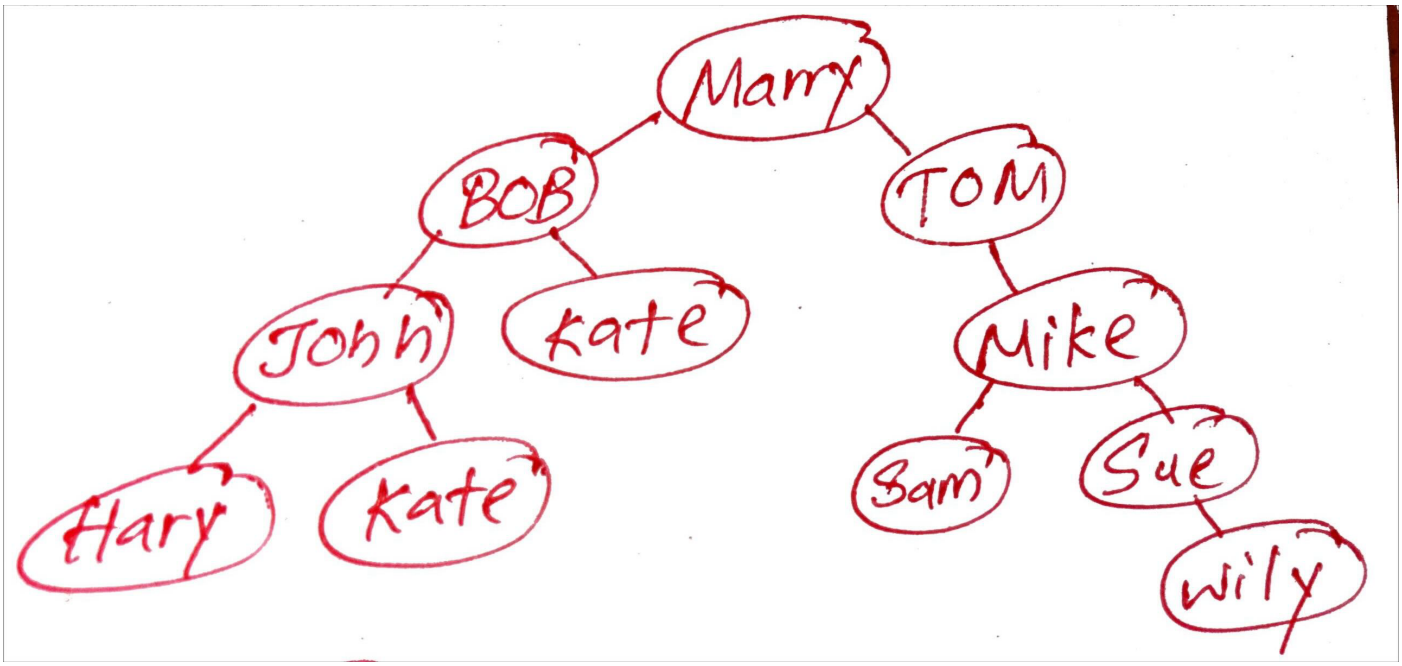
iii. 120,46,7,2,31,39,23,40



iv. S, K, R, B, C, M, W, A, Z, X, D



v. Mary, Tom, Bob, Wily, John, Mike, Harry, Kate, Sam, Sue



Question 01. B).

For Integers,

```
class IntNode {
    int data;
    IntNode left, right;

    public IntNode(int data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class IntBST {
    IntNode root;

    public void insert(int data) {
        root = insertRec(root, data);
    }

    private IntNode insertRec(IntNode node, int data) {
        if (node == null) return new IntNode(data);

        if (data < node.data)
            node.left = insertRec(node.left, data);
        else if (data > node.data)
            node.right = insertRec(node.right, data);

        return node;
    }
```

```
    public void preorder(IntNode node) {
        if (node != null) {
            System.out.print(node.data + " ");
            preorder(node.left);
            preorder(node.right);
        }
    }

    public void inorder(IntNode node) {
        if (node != null) {
            inorder(node.left);
            System.out.print(node.data + " ");
            inorder(node.right);
        }
    }

    public void postorder(IntNode node) {
        if (node != null) {
            postorder(node.left);
            postorder(node.right);
            System.out.print(node.data + " ");
        }
    }
}
```

For Character,

```
class CharNode {
    char data;
    CharNode left, right;

    public CharNode(char data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class CharBST {
    CharNode root;

    public void insert(char data) {
        root = insertRec(root, data);
    }

    private CharNode insertRec(CharNode node, char data) {
        if (node == null) return new CharNode(data);

        if (data < node.data)
            node.left = insertRec(node.left, data);
        else if (data > node.data)
            node.right = insertRec(node.right, data);

        return node;
    }

    public void preorder(CharNode node) {
```

```
        if (node != null) {
            System.out.print(node.data + " ");
            preorder(node.left);
            preorder(node.right);
        }
    }

    public void inorder(CharNode node) {
        if (node != null) {
            inorder(node.left);
            System.out.print(node.data + " ");
            inorder(node.right);
        }
    }

    public void postorder(CharNode node) {
        if (node != null) {
            postorder(node.left);
            postorder(node.right);
            System.out.print(node.data + " ");
        }
    }
}
```

For Strings,

```
class StrNode {
    String data;
    StrNode left, right;

    public StrNode(String data) {
        this.data = data;
        left = null;
        right = null;
    }
}

class StrBST {
    StrNode root;

    public void insert(String data) {
        root = insertRec(root, data);
    }

    private StrNode insertRec(StrNode node, String data) {
        if (node == null) return new StrNode(data);

        if (data.compareTo(node.data) < 0)
            node.left = insertRec(node.left, data);
        else if (data.compareTo(node.data) > 0)
            node.right = insertRec(node.right, data);

        return node;
    }

    public void preorder(StrNode node) {
```

```
        if (node != null) {
            System.out.print(node.data + " ");
            preorder(node.left);
            preorder(node.right);
        }
    }

    public void inorder(StrNode node) {
        if (node != null) {
            inorder(node.left);
            System.out.print(node.data + " ");
            inorder(node.right);
        }
    }

    public void postorder(StrNode node) {
        if (node != null) {
            postorder(node.left);
            postorder(node.right);
            System.out.print(node.data + " ");
        }
    }
}
```

```
public class PracticalTutorial06 {
    public static void main(String[] args) {

        int[] seq1 = {69, 25, 62, 88, 10, 37, 99};
        IntBST bst1 = new IntBST();
        for (int num : seq1) bst1.insert(num);
        System.out.println("Sequence i:");
        System.out.print("Preorder: ");
        bst1.preorder(bst1.root); System.out.println();
        System.out.print("Inorder: ");
        bst1.inorder(bst1.root); System.out.println();
        System.out.print("Postorder: ");
        bst1.postorder(bst1.root); System.out.println("\n");

        int[] seq2 = {32, 24, 6, 2, 39, 41, 30, 145};
        IntBST bst2 = new IntBST();
        for (int num : seq2) bst2.insert(num);
        System.out.println("Sequence ii:");
        System.out.print("Preorder: ");
        bst2.preorder(bst2.root); System.out.println();
        System.out.print("Inorder: ");
        bst2.inorder(bst2.root); System.out.println();
        System.out.print("Postorder: ");
        bst2.postorder(bst2.root); System.out.println("\n");
```

```
        int[] seq3 = {120, 46, 7, 2, 31, 39, 23, 40};
        IntBST bst3 = new IntBST();
        for (int num : seq3) bst3.insert(num);
        System.out.println("Sequence iii:");
        System.out.print("Preorder: ");
        bst3.preorder(bst3.root); System.out.println();
        System.out.print("Inorder: ");
        bst3.inorder(bst3.root); System.out.println();
        System.out.print("Postorder: ");
        bst3.postorder(bst3.root); System.out.println("\n");


        char[] seq4 = {'S', 'K', 'R', 'B', 'C', 'M', 'W', 'A', 'Z', 'X', 'D'};
        CharBST bst4 = new CharBST();
        for (char ch : seq4) bst4.insert(ch);
        System.out.println("Sequence iv:");
        System.out.print("Preorder: ");
        bst4.preorder(bst4.root); System.out.println();
        System.out.print("Inorder: ");
        bst4.inorder(bst4.root); System.out.println();
        System.out.print("Postorder: ");
        bst4.postorder(bst4.root); System.out.println("\n");
```

```
        String[] seq5 = {"Mary", "Tom", "Bob", "Wily", "John", "Mike", "Harry",
"Kate", "Sam", "Sue"};

        StrBST bst5 = new StrBST();

        for (String str : seq5) bst5.insert(str);

        System.out.println("Sequence v:");

        System.out.print("Preorder: ");

        bst5.preorder(bst5.root); System.out.println();

        System.out.print("Inorder: ");

        bst5.inorder(bst5.root); System.out.println();

        System.out.print("Postorder: ");

        bst5.postorder(bst5.root); System.out.println();

    }

}
```

```
 Introduction to Organizational Behavior.pdf            St
 Main.java
kavindus@kavindus-MacBook-Air BECS 21722 - Organizational
 Sequence i:
 Preorder: 69 25 10 62 37 88 99
 Inorder: 10 25 37 62 69 88 99
 Postorder: 10 37 62 25 99 88 69

 Sequence ii:
 Preorder: 32 24 6 2 30 39 41 145
 Inorder: 2 6 24 30 32 39 41 145
 Postorder: 2 6 30 24 145 41 39 32

 Sequence iii:
 Preorder: 120 46 7 2 31 23 39 40
 Inorder: 2 7 23 31 39 40 46 120
 Postorder: 2 23 40 39 31 7 46 120

 Sequence iv:
 Preorder: S K B A C D R M W Z X
 Inorder: A B C D K M R S W X Z
 Postorder: A D C B M R K X Z W S

 Sequence v:
 Preorder: Mary Bob John Harry Kate Tom Mike Sam Sue Wily
 Inorder: Bob Harry John Kate Mary Mike Sam Sue Tom Wily
 Postorder: Harry Kate John Bob Sue Sam Mike Wily Tom Mary
kavindus@kavindus-MacBook-Air BECS 21722 - Organizational
```