**Assignment Of Data Structure**

****

**Submitted To:** **Submitted By:**

Prof. Rajender Lakshay

Singh Chiller Khanagwal

**Roll No:**

21128

**Wright a Program On Linear Search**

public class l\_search {

public static int linearSearch(int[] arr, int key)

{

for(int i=0;i<arr.length;i++){

if(arr[i] == key){

return i;

}

}

return -1;

}

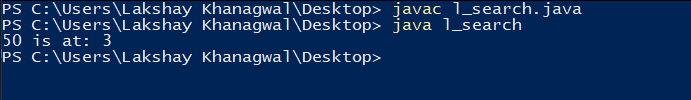
public static void main(String a[]){

int[] a1= {10,20,30,50,70,90};

int key = 50;

System.out.println(key+" is at: "+linearSearch(a1, key));

}

} 

**Wright a Program On Binary Search**

class b\_search {

int b\_Search(int arr[], int l, int r, int x)

{

if (r >= l) {

int mid = l + (r - l) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] > x)

return b\_Search(arr, l, mid - 1, x);

return b\_Search(arr, mid + 1, r, x);

}

return -1;

}

public static void main(String args[])

{

b\_search ob = new b\_search();

int arr[] = { 2, 3, 4, 10, 40 };

int n = arr.length;

int x = 10;

int result = ob.b\_Search(arr, 0, n - 1, x);

if (result == -1)

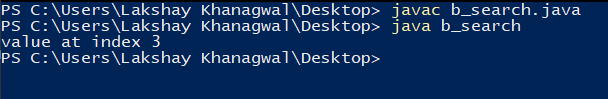
System.out.println("value not present");

else

System.out.println("value at index "+ result);

}

}



**Wright a Program On Inorder**

class Node {

int key;

Node left, right;

public Node(int item)

{

key = item;

left = right = null;

}

}

class in\_order {

Node root;

in\_order() { root = null; }

void printInorder(Node node)

{

if (node == null)

return;

printInorder(node.left);

System.out.print(node.key + " ");

printInorder(node.right);

}

void printInorder() { printInorder(root); }

public static void main(String[] args)

{

in\_order tree = new in\_order();

tree.root = new Node(1);

tree.root.left = new Node(2);

tree.root.right = new Node(3);

tree.root.left.left = new Node(4);

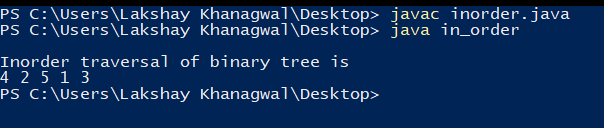
tree.root.left.right = new Node(5);

System.out.println("\nInorder traversal of binary tree is ");

tree.printInorder();

}

}



**Wright a Program On Preorder**

class Node {

int key;

Node left, right;

public Node(int item)

{

key = item;

left = right = null;

}

}

class pre\_order {

Node root;

pre\_order() { root = null; }

void printPreorder(Node node)

{

if (node == null)

return;

System.out.print(node.key + " ");

printPreorder(node.left);

printPreorder(node.right);

}

void printPreorder() { printPreorder(root); }

public static void main(String[] args)

{

pre\_order tree = new pre\_order();

tree.root = new Node(1);

tree.root.left = new Node(2);

tree.root.right = new Node(3);

tree.root.left.left = new Node(4);

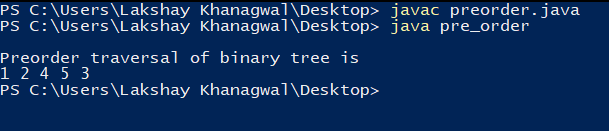
tree.root.left.right = new Node(5);

System.out.println("\nPreorder traversal of binary tree is ");

tree.printPreorder();

}

}



**Wright a Program On Postorder**

class Node {

int key;

Node left, right;

public Node(int item)

{

key = item;

left = right = null;

}

}

class pos\_order {

Node root;

pos\_order() { root = null; }

void printPostorder(Node node)

{

if (node == null)

return;

printPostorder(node.left);

printPostorder(node.right);

System.out.print(node.key + " ");

}

void printPostorder() { printPostorder(root); }

public static void main(String[] args)

{

pos\_order tree = new pos\_order();

tree.root = new Node(1);

tree.root.left = new Node(2);

tree.root.right = new Node(3);

tree.root.left.left = new Node(4);

tree.root.left.right = new Node(5);

System.out.println("\nPostorder traversal of binary tree is ");

tree.printPostorder();

}

}

