Name:Yogini Kale
Roll No : SECOA155
Class: A

ASSIGNMENT NO. 5

AIM: Write a program to create a binary tree if preorder and inorder traversals are given in java.

INDEX TERMS:array,binary search tree.

THEORY: Java is a computer programming language that is concurrent,classbased and object-oriented.it is platform independent.the two basic keywords are used here which are public and private.keyword public denotes that method can be called from code in other class unlike private whose members cannot be accessed any where .also static is one of the term used in java.keyword static in front of method indicates static method,which is associatedonly with class and not with any specific instance of that class.

.

Mathematical model: -we can define binary tree using set as follows:
a tree T={V,E}wherev={V1,V2,.,Vn}where n is no.of vertices.
E={E1,E2,..,En}such that |E|=|V|-1
Eij={Vi,Vj}such that j=2*I or (2*i)=1
Insert V2 to V1
V=NULL,E=NULL
After insertion of V2 to V1,V=V U{V2},E=E U{E12}
Probability of insertion of V2 to V1 will be p(V2|V1)=P(V2∩ V1) | P(V1)
Delete V2S After deletion of V2,V=V-{V2}.If V2 is child of V1,E=E-{E1}
INPUTS:- Set of numbers,characters or string to create binary tree.
OUTPUT:- Set of numbers,characters or string displayed in inorder or preorder or postordr.

PROGRAM:-

```
 import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class TNode
{
```

```
 int data;
 TNode left;
 TNode right;

 TNode(int d)
        {
  data = d;
  left= null;
  right=null;
       }
}

class TreeBuilder
{
 static int preIndex;
 static int[] in, pre;
 TNode Start;

 static void setValue(int[] i, int[] p)
      {
  in = i;
  pre = p;
  preIndex=0;
     }

 static int findInIndex(int inStart, int inEnd, int value)
     {
  for(int i=inStart; i<=inEnd; i++)
   if(in[i]==value)
   return i;

  return -1;
         }


 static TNode buildTree(int inStart, int inEnd)
      {
  if(inStart>inEnd)
   return null;

  TNode node = new TNode(pre[preIndex++]);
```

```
  if(inStart==inEnd)
   return node;

  int inIndex = findInIndex(inStart, inEnd, node.data);

  node.left=buildTree(inStart, inIndex-1);
  node.right=buildTree(inIndex+1, inEnd);

  return node;
       }
}

class TreeBuilderDemo
 {
  public static void main(String[] args)
           {
  int i,j;

     System.out.print("\nEnter Inorder Sequence of Tree");

     int[] in = {0, 0, 0, 0, 0};

     for(i = 0 ; i<5; i++)
          {
      BufferedReader br = new BufferedReader( new InputStreamReader(System.in));
           try
                  {
                 in[i]  =      Integer.parseInt(br.readLine());
                  }
        catch (NumberFormatException e)
                 {
       e.printStackTrace();
                  }
    catch (IOException e)
                  {
       e.printStackTrace();
                  }
      }
      System.out.print("\nEnter Preorrder Sequence of Tree");
```

```
  int[] pre = {0, 0, 0, 0, 0};
   for(j = 0 ; j<5; j++)
     {
BufferedReader br = new BufferedReader(
                   new InputStreamReader(System.in));
   try
{
    pre[j]  = Integer.parseInt(br.readLine());
}
catch (NumberFormatException e)
  {
     e.printStackTrace();
}
catch (IOException e)
 {
     e.printStackTrace();
}
   }

  TreeBuilder.setValue(in, pre);
 TNode start = TreeBuilder.buildTree(0,in.length-1);


  System.out.print("\nPreorder\t: ");
  printPreorder(start)
  System.out.print("\n\nInorder\t\t: ");
  printInorder(start);

  System.out.print("\n\nPostorder\t: ");
  printPostorder(start);

  System.out.println("");
 }

 static void printInorder(TNode node)
     {
  if (node == null)
   return;
  printInorder(node.left);
   System.out.print(node.data + "\t ");
   printInorder(node.right);
```

```
    }
 static void printPreorder(TNode node)
  {
   if (node == null)
    return;
            System.out.print(node.data + "\t ");
   printPreorder(node.left);
   printPreorder(node.right);
  }
             static void printPostorder(TNode node)
  {
                 if (node == null)
    return;
                          printPostorder(node.left);
                         printPostorder(node.right);
                        System.out.print(node.data + "\t ");
 }
}
```

CONCLUSION:- Thus, we have created binary tree if preorder and inorder is given using JAVA programming.