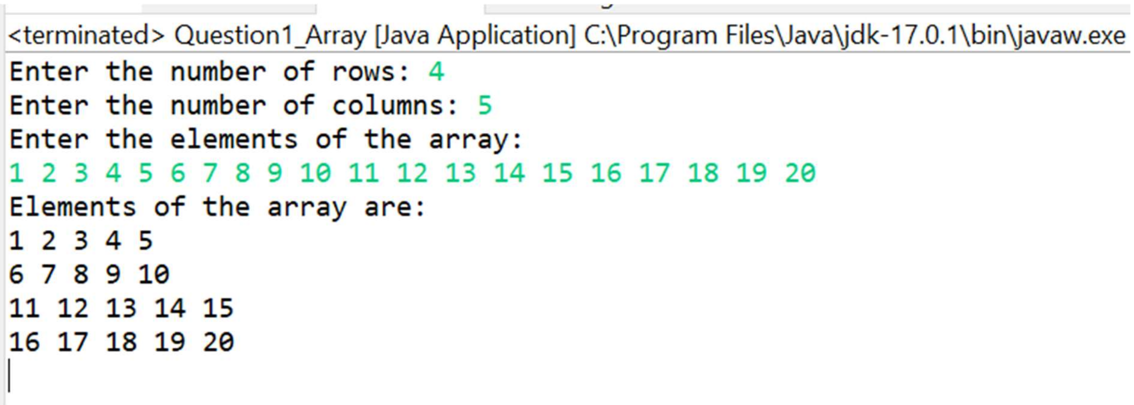
CSE2001 (Data Structures & Algorithms) Lab-6

KHAN MOHD OWAIS RAZA
20BCD7138

1. Get 20 numbers from user and store in array. Create a Binary search tree in the sequence of input. Perform the following:
(i) Insert an element into BST.
(ii) Delete an element from BST.
(iii) Search an element in BST.

Java code to get 20 numbers from user and store in array:

```java
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-6 (15-10-2022)*/
/* Java code to get 20 numbers from user and store in array */
package CSE2001_Lab6_20BCD7138;
import java.util.Scanner;
public class Question1_Array{
public static void main(String args[]){
int m, n, i, j;
Scanner sc=new Scanner(System.in);
System.out.print("Enter the number of rows: ");
m = sc.nextInt();
System.out.print("Enter the number of columns: ");
n = sc.nextInt();
int array[][] = new int[m][n];
System.out.println("Enter the elements of the array: ");
for (i = 0; i < m; i++)
for (j = 0; j < n; j++)
array[i][j] = sc.nextInt();
System.out.println("Elements of the array are: ");
for (i = 0; i < m; i++){
for (j = 0; j < n; j++)
System.out.print(array[i][j] + " ");
System.out.println();
}}}
```



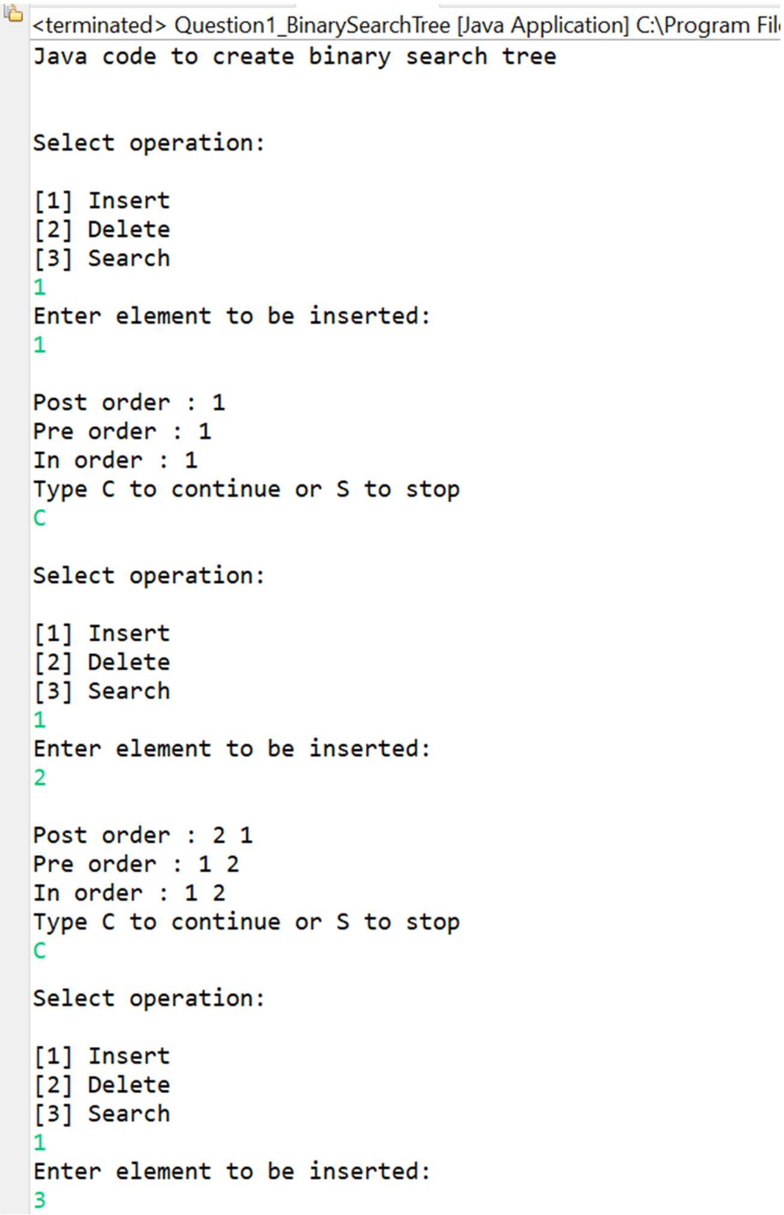
<terminated> Question1_Array [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe

```
Enter the number of rows: 4
Enter the number of columns: 5
Enter the elements of the array:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Elements of the array are:
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
```

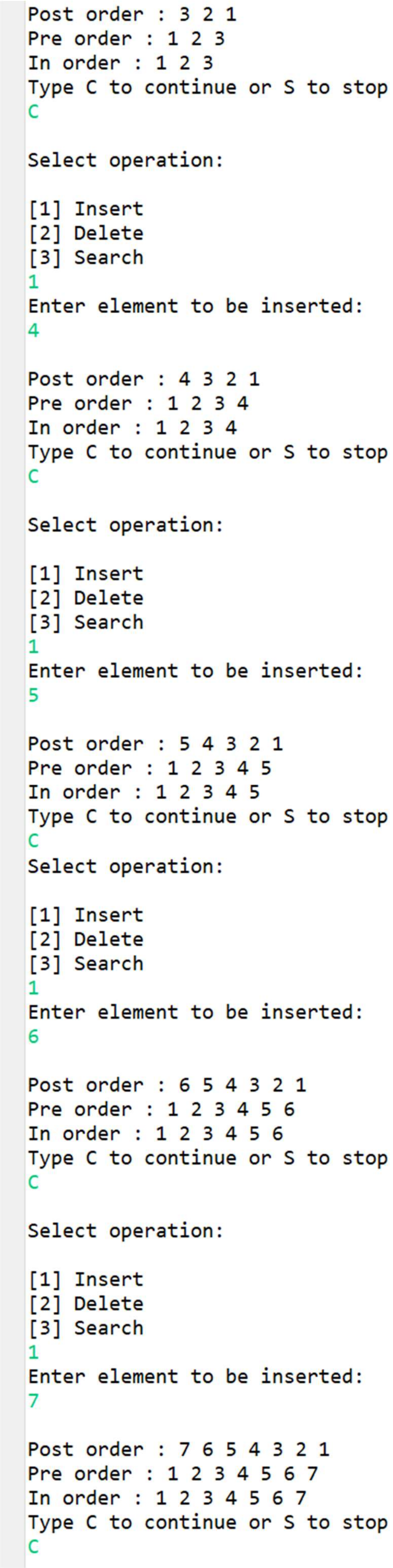
Java code to create binary search tree:

```
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-6 (15-10-2022)*/
/* Java code for binary search tree and operations */
package CSE2001_Lab6_20BCD7138;
import java.util.Scanner;
class BSTNode{
BSTNode left, right;
int data;
public BSTNode(){
left = null;
right = null;
data = 0;
}
public BSTNode(int n){
left = null;
right = null;
data = n;
}
public void setLeft(BSTNode n){
left = n;
}
public void setRight(BSTNode n){
right = n;
}
public BSTNode getLeft(){
return left;
}
public BSTNode getRight(){
return right;
}
public void setData(int d){
data = d;
}
public int getData(){
return data;
}}
class BST{
private BSTNode root;
public BST(){
root = null;
}
public boolean isEmpty(){
return root == null;
}
public void insert(int data){
root = insert(root, data);
}
private BSTNode insert(BSTNode node, int data){
if (node == null) node = new BSTNode(data);
else{
if (data <= node.getData()) node.left = insert(node.left, data);
else node.right = insert(node.right, data);
}
return node;
}
public void delete(int k){
if (isEmpty()) System.out.println("Tree Empty");
else if (search(k) == false) System.out.println("Sorry "+ k +" is not present");
else{
root = delete(root, k);
System.out.println(k+ " deleted from the tree");
}}
private BSTNode delete(BSTNode root, int k){
BSTNode p, p2, n;
if (root.getData() == k){
BSTNode lt, rt;
lt = root.getLeft();
rt = root.getRight();
if (lt == null && rt == null) return null;
else if (lt == null){
p = rt;
return p;
}
else if (rt == null){
p = lt;
return p;
}
else{
p2 = rt;
p = rt;
while (p.getLeft() != null) p = p.getLeft();
p.setLeft(lt);
return p2;
}}
if (k < root.getData()){
n = delete(root.getLeft(), k);
root.setLeft(n);
}
else{
n = delete(root.getRight(), k);
root.setRight(n);
}
return root;
}
public int countNodes(){
return countNodes(root);
}
private int countNodes(BSTNode r){
if (r == null) return 0;
else{
int l = 1;
l += countNodes(r.getLeft());
l += countNodes(r.getRight());
return l;
}}
public boolean search(int val){
return search(root, val);
}
private boolean search(BSTNode r, int val){
boolean found = false;
```

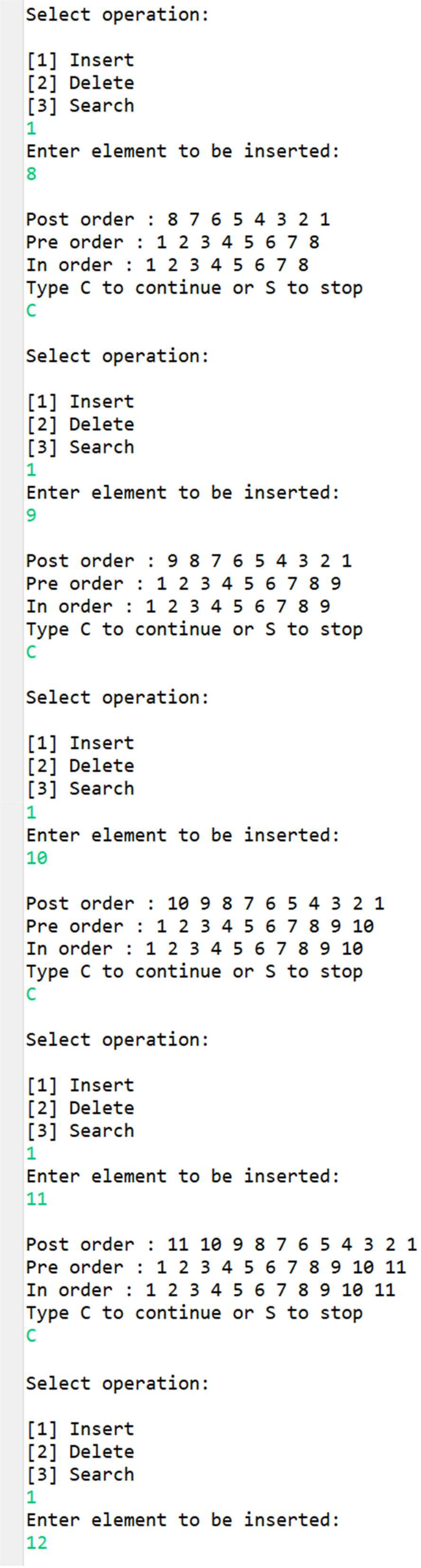
```java
while ((r != null) && !found){
int rval = r.getData();
if (val < rval) r = r.getLeft();
else if (val > rval) r = r.getRight();
else{
found = true;
break;
}
found = search(r, val);
}
return found;
}
public void inorder(){
inorder(root);
}
private void inorder(BSTNode r){
if (r != null){
inorder(r.getLeft());
System.out.print(r.getData() +" ");
inorder(r.getRight());
}}
public void preorder(){
preorder(root);
}
private void preorder(BSTNode r){
if (r != null){
System.out.print(r.getData() +" ");
preorder(r.getLeft());
preorder(r.getRight());
}}
public void postorder(){
postorder(root);
}
private void postorder(BSTNode r){
if (r != null){
postorder(r.getLeft());
postorder(r.getRight());
System.out.print(r.getData() +" ");
}}}
public class Question1_BinarySearchTree{
public static void main(String[] args){
Scanner scan = new Scanner(System.in);
BST bst = new BST();
System.out.println("Java code to create binary search tree\n");
char ch;
do{
System.out.println("\nSelect operation: \n");
System.out.println("[1] Insert");
System.out.println("[2] Delete");
System.out.println("[3] Search");
int choice = scan.nextInt();
switch (choice){
case 1 :
System.out.println("Enter element to be inserted:");
bst.insert( scan.nextInt() );
break;
case 2 :
System.out.println("Enter element to be inserted:");
bst.delete( scan.nextInt() );
break;
case 3 :
System.out.println("Enter element to be searched:");
System.out.println("Search result : "+ bst.search( scan.nextInt() ));
break;
}
System.out.print("\nPost order : ");
bst.postorder();
System.out.print("\nPre order : ");
bst.preorder();
System.out.print("\nIn order : ");
bst.inorder();
System.out.println("\nType C to continue or S to stop");
ch = scan.next().charAt(0);
} while (ch == 'C');
}}
```



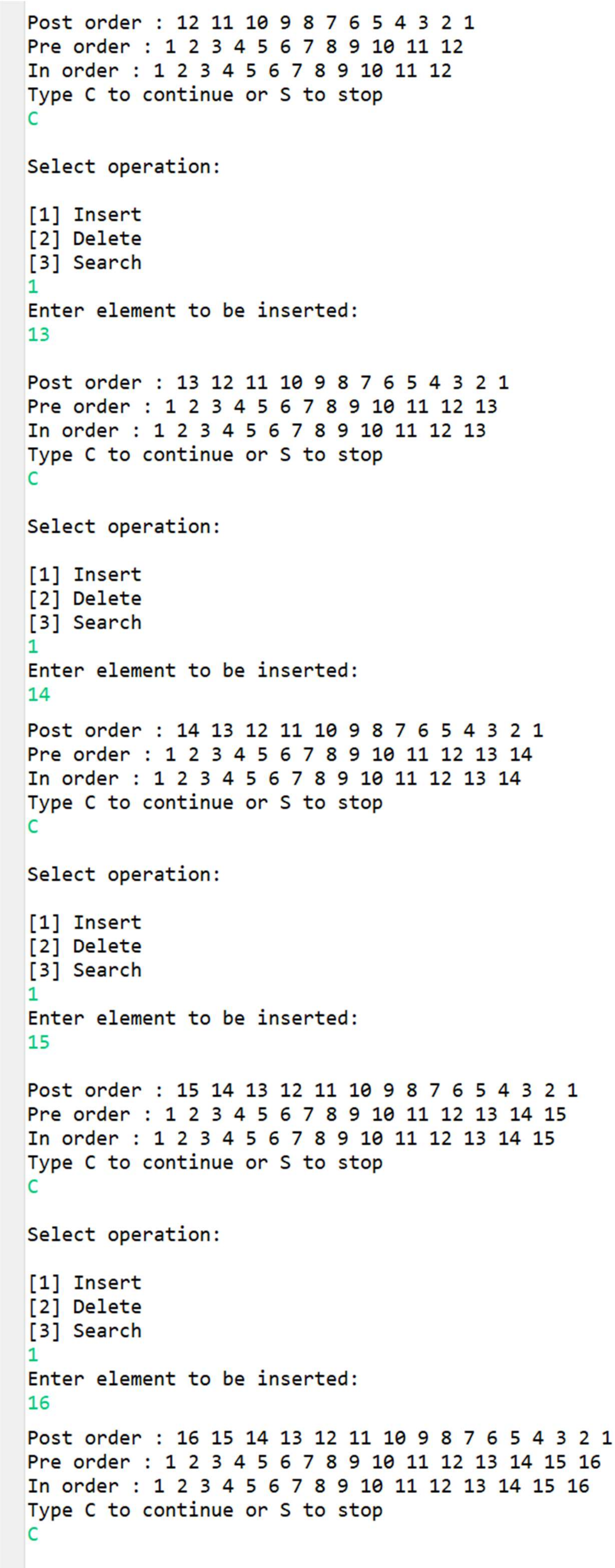
```
<terminated> Question1_BinarySearchTree [Java Application] C:\Program Fil
Java code to create binary search tree


Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
1

Post order : 1
Pre order : 1
In order : 1
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
2

Post order : 2 1
Pre order : 1 2
In order : 1 2
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
3
```



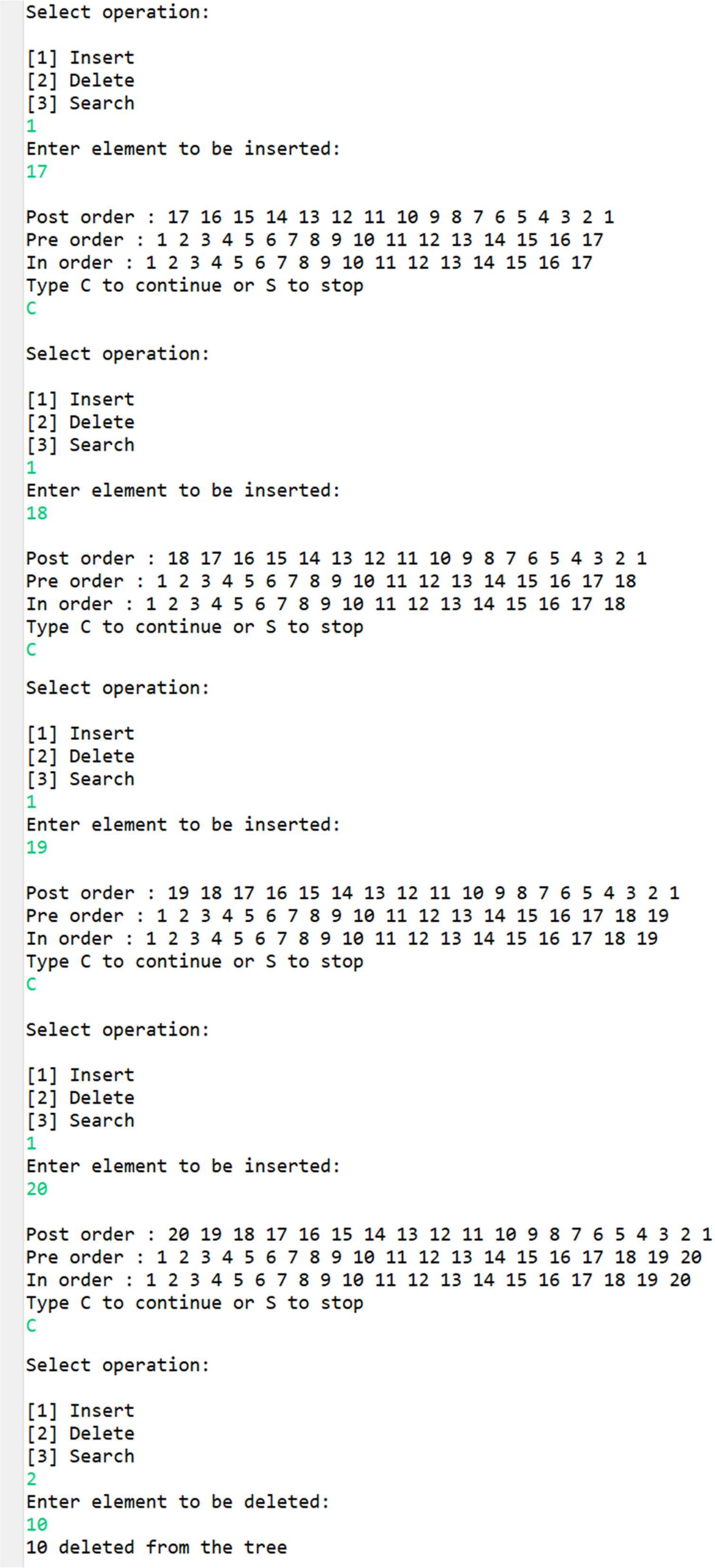
```
Post order : 3 2 1
Pre order : 1 2 3
In order : 1 2 3
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
4

Post order : 4 3 2 1
Pre order : 1 2 3 4
In order : 1 2 3 4
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
5

Post order : 5 4 3 2 1
Pre order : 1 2 3 4 5
In order : 1 2 3 4 5
Type C to continue or S to stop
C
Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
6

Post order : 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6
In order : 1 2 3 4 5 6
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
7

Post order : 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7
In order : 1 2 3 4 5 6 7
Type C to continue or S to stop
C
```



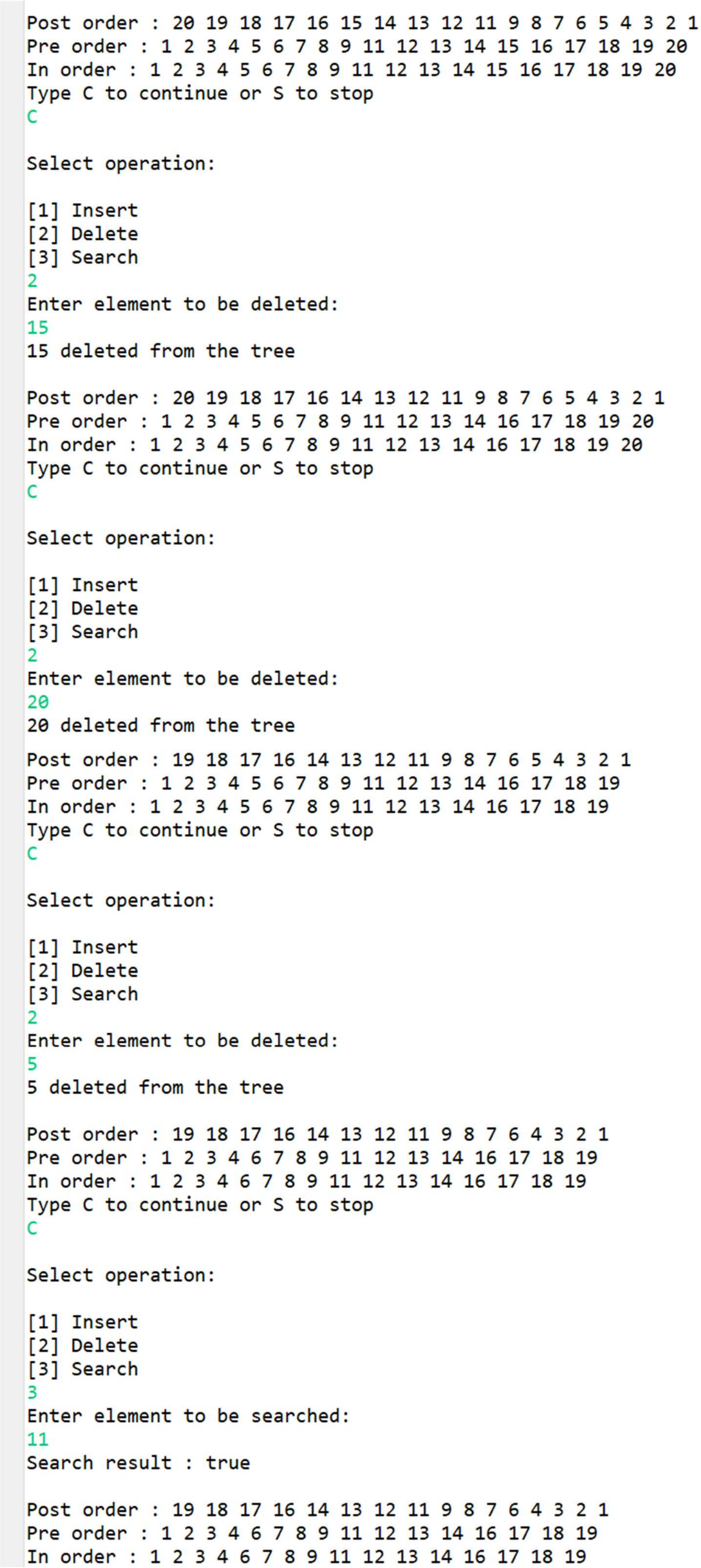
```
Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
8

Post order : 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8
In order : 1 2 3 4 5 6 7 8
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
9

Post order : 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9
In order : 1 2 3 4 5 6 7 8 9
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
10

Post order : 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10
In order : 1 2 3 4 5 6 7 8 9 10
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
11

Post order : 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11
In order : 1 2 3 4 5 6 7 8 9 10 11
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
12
```



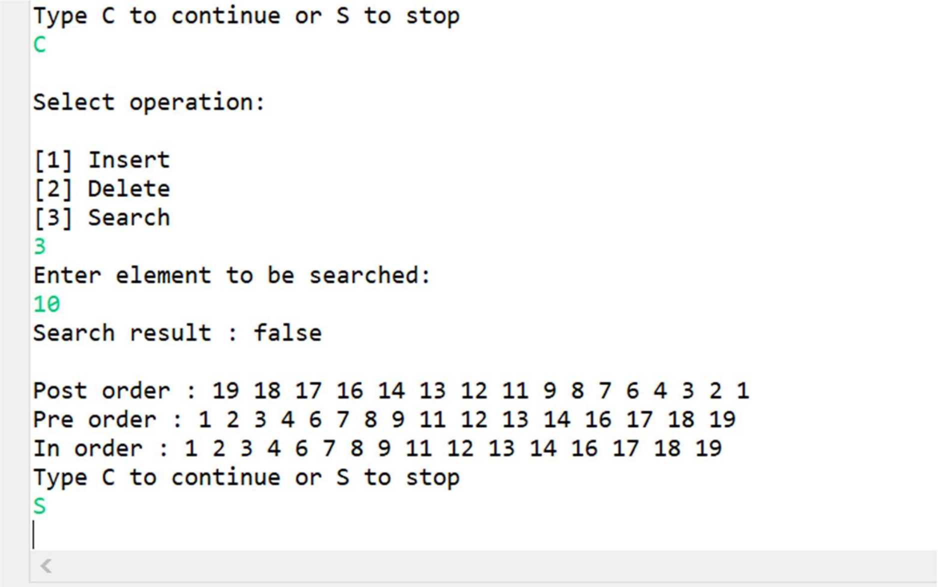
```
Post order : 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12
In order : 1 2 3 4 5 6 7 8 9 10 11 12
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
13

Post order : 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
14

Post order : 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
15

Post order : 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
16

Post order : 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Type C to continue or S to stop
C
```



```
Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
17

Post order : 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
18

Post order : 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
19

Post order : 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
1
Enter element to be inserted:
20

Post order : 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
In order : 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
2
Enter element to be deleted:
10
10 deleted from the tree
```



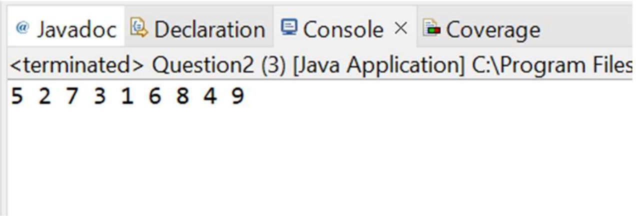
```
Post order : 20 19 18 17 16 15 14 13 12 11 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20
In order : 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
2
Enter element to be deleted:
15
15 deleted from the tree

Post order : 20 19 18 17 16 14 13 12 11 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 11 12 13 14 16 17 18 19 20
In order : 1 2 3 4 5 6 7 8 9 11 12 13 14 16 17 18 19 20
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
2
Enter element to be deleted:
20
20 deleted from the tree
Post order : 19 18 17 16 14 13 12 11 9 8 7 6 5 4 3 2 1
Pre order : 1 2 3 4 5 6 7 8 9 11 12 13 14 16 17 18 19
In order : 1 2 3 4 5 6 7 8 9 11 12 13 14 16 17 18 19
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
2
Enter element to be deleted:
5
5 deleted from the tree

Post order : 19 18 17 16 14 13 12 11 9 8 7 6 4 3 2 1
Pre order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
In order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
3
Enter element to be searched:
11
Search result : true

Post order : 19 18 17 16 14 13 12 11 9 8 7 6 4 3 2 1
Pre order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
In order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
```



```
Type C to continue or S to stop
C

Select operation:

[1] Insert
[2] Delete
[3] Search
3
Enter element to be searched:
10
Search result : false

Post order : 19 18 17 16 14 13 12 11 9 8 7 6 4 3 2 1
Pre order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
In order : 1 2 3 4 6 7 8 9 11 12 13 14 16 17 18 19
Type C to continue or S to stop
S
```

2. Create AVL Tree (Balanced BST) for the following sequence 3,2,1,4,5,6,7,8,9

```java
/**
Name: KHAN MOHD OWAIS RAZA
ID : 20BCD7138
Course: Data Structures & Algorithm
Code: CSE2001
Slot: L19+L20
**/
/* Lab-6 (15-10-2022)*/
/* java code for AVL tree*/
package CSE2001_Lab6_20BCD7138;
import java.util.*;
public class Question2{
static class TNode {
int data;
TNode left;
TNode right;
}
static TNode sortedArrayToBST(int arr[], int start, int end){
if (start > end) return null;
int mid = (start + end) / 2;
TNode root = newNode(arr[mid]);
root.left = sortedArrayToBST(arr, start, mid - 1);
root.right = sortedArrayToBST(arr, mid + 1, end);
return root;
}
static TNode newNode(int data){
TNode node = new TNode();
node.data = data;
node.left = null;
node.right = null;
return node;
}
static void printLevelOrder(TNode root){
if (root == null) return;
Queue<TNode > q= new LinkedList<TNode>();
q.add(root);
while (q.size()>0){
TNode node = q.element();
System.out.print( node.data + " ");
q.remove();
if (node.left != null) q.add(node.left);
if (node.right != null) q.add(node.right);
}}
public static void main(String args[]){
int arr[] = {3,2,1,4,5,6,7,8,9};
int n = arr.length;
TNode root = sortedArrayToBST(arr, 0, n - 1);
printLevelOrder(root);
}}
```



```
<terminated> Question2 (3) [Java Application] C:\Program Files
5 2 7 3 1 6 8 4 9
```