Sukkur IBA University
Department of Computer Science

DATA STRUCTURES
Lab07 – Complete Binary Tree , Priority Queues

Instructor: Saif Hassan

READ IT FIRST

Prior to start solving the problems in this assignments, please give full concentration on following points.

1. WORKING – This is individual lab. If you are stuck in a problem contact your teacher, but, in mean time start doing next question (don't waste time).
2. DEADLINE – 11th March, 2022
3. SUBMISSION – This assignment needs to be submitted in a soft copy.
4. WHERE TO SUBMIT – Please visit your LMS.
5. WHAT TO SUBMIT – Submit this docx and pdf file.

KEEP IT WITH YOU!

1. Indent your code inside the classes and functions. It's a good practice!
2. It is not bad if you keep your code indented inside the loops, if and else blocks as well.
3. Comment your code, where it is necessary.
4. Read the entire question. Don't jump to the formula directly.

I, __Amjad Ali_____ with student ID _191-21-0001__

Section __A__hereby declare that I do understand the instructions above and follow them. This is my own work.

Exercises

Task1 Description

Task 01: (Priority Queue using unsorted array)

A **PriorityQueue** is used when the objects are supposed to be processed based on the priority. It is known that a queue follows First-In-First-Out algorithm, but sometimes the elements of the queue are needed to be processed according to the priority, that's when the **PriorityQueue** comes into play. The **PriorityQueue** is based on the priority heap. The elements of the priority queue are ordered according to the natural ordering, or by a Comparator provided at queue construction time, depending on which constructor is used.

You have been provided file named **"PriorityQueueUsingArray.java"**. Your task is to complete **methods** within **PriorityQueueUsingArray** class.

In this case, running time for insert will be $O(1)$ and for extractMax/getMax will be $O(n)$.

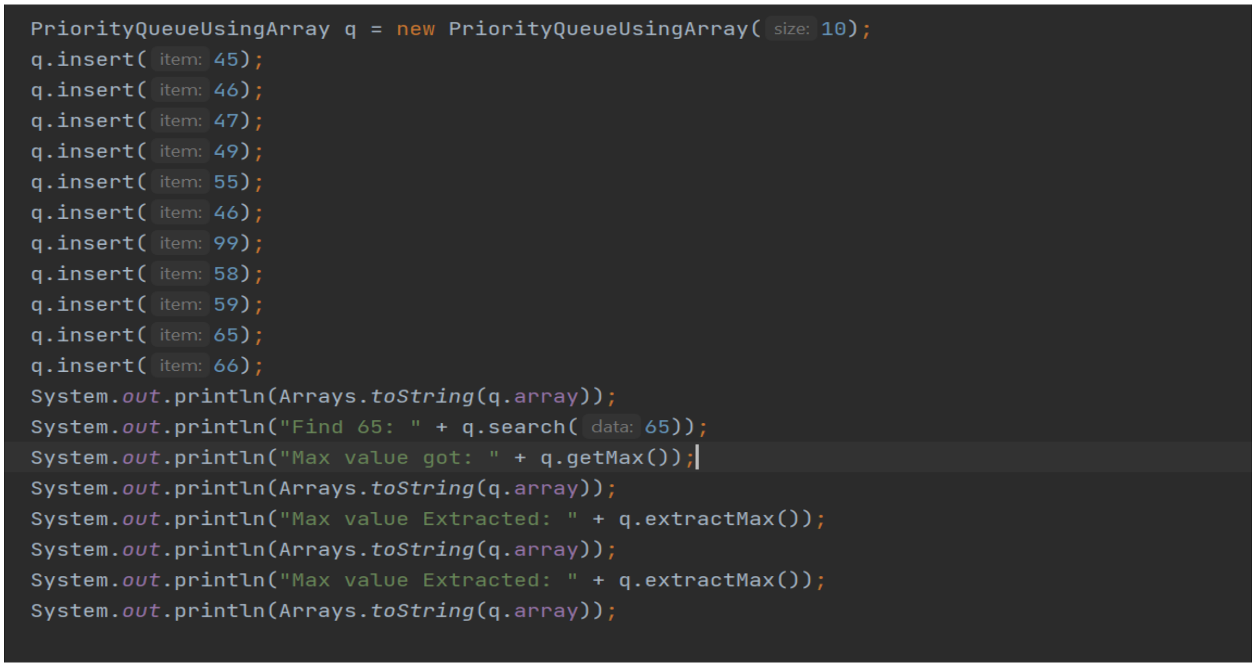
Solution:

```
import java.util.Arrays;

public class PriorityQueueUsingArray {
    //unsorted Array
    private int[] array;
    private int rear = -1;
    private int size;

    public PriorityQueueUsingArray(int size) {
         this.size = size;
         array = new int[size];
     }

     public void insert(int item) {
         if (rear + 1 == size) {
             System.out.println("Queue is overflow");
         } else {
             array[++rear] = item;
             System.out.println("Successfully added!");
         }

     }

     public int getMax() {
         if (rear == -1) {
             System.out.println("Queue is Underflow");
             return -1;
         }
         int max = 0;
```

```
        for (int i = 0; i <= rear; i++) {
            max = Math.max(max, array[i]);
        }

        return max;

    }

    public int extractMax() {
        if (rear == -1) {
            System.out.println("Queue is Underflow");
            return -1;
        }
        int max = 0;
        int ind = 0;
        for (int i = 0; i <= rear; i++) {
            if (max < array[i]) {
                max = array[i];
                ind = i;
            }
        }
        for (int i = ind; i < rear; i++)
            array[i] = array[i + 1];

        array[rear] = 0;
        rear--;


        return max;
    }

    public boolean search(int data) {
        if (rear == -1) {
            System.out.println("Queue is Underflow");
            return false;
        }

        for (int i = 0; i <= rear; i++) {
            if (data == array[i]) {
                return true;
            }
        }

        return false;


    }

    public static void main(String[] args) {

```

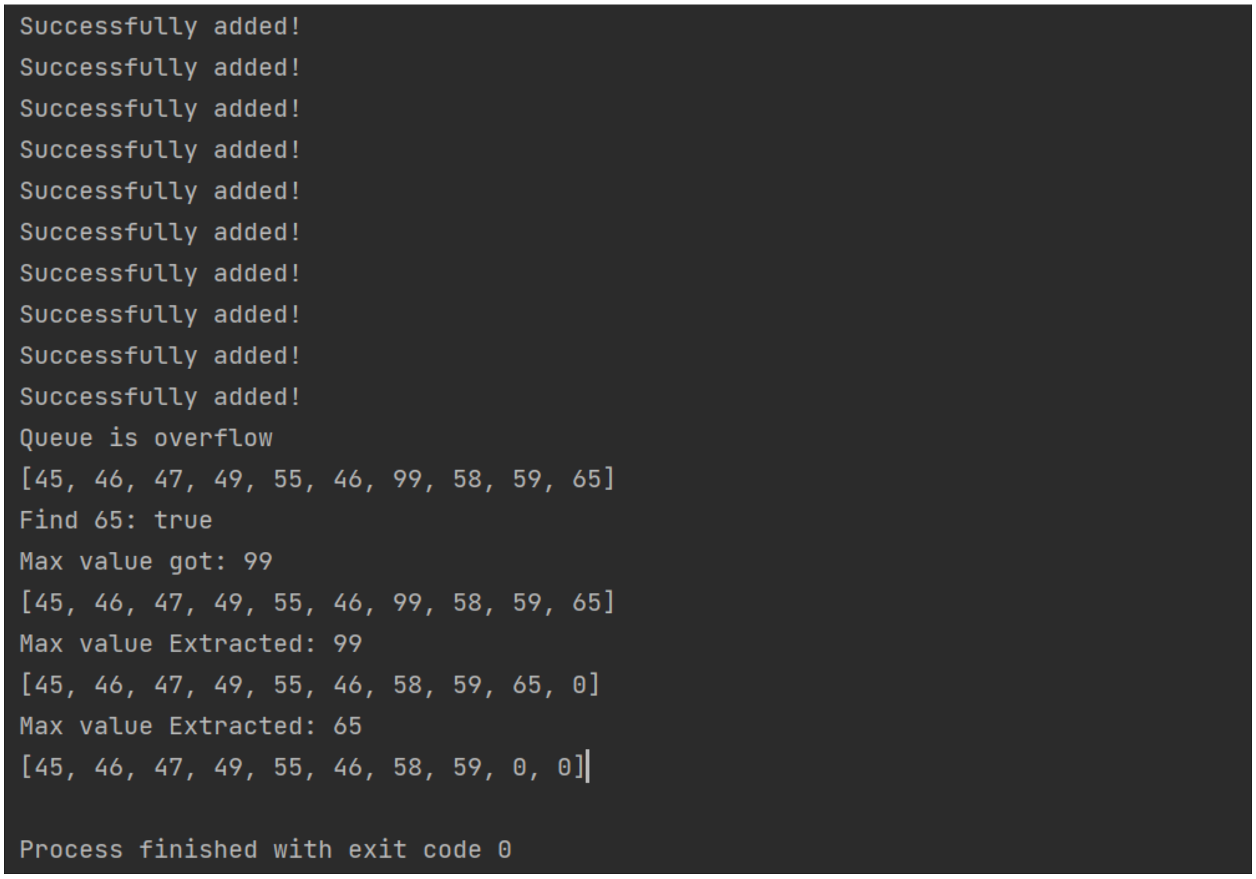
```java
        PriorityQueueUsingArray q = new PriorityQueueUsingArray(10);
        q.insert(45);
        q.insert(46);
        q.insert(47);
        q.insert(49);
        q.insert(55);
        q.insert(46);
        q.insert(99);
        q.insert(58);
        q.insert(59);
        q.insert(65);
        q.insert(66);
        System.out.println(Arrays.toString(q.array));
        System.out.println("Find 65: " + q.search(65));
        System.out.println("Max value got: " + q.getMax());
        System.out.println(Arrays.toString(q.array));
        System.out.println("Max value Extracted: " + q.extractMax());
        System.out.println(Arrays.toString(q.array));
        System.out.println("Max value Extracted: " + q.extractMax());
        System.out.println(Arrays.toString(q.array));


    }

}
```

<u>Sample Input:</u>



```java
PriorityQueueUsingArray q = new PriorityQueueUsingArray( size: 10);
q.insert( item: 45);
q.insert( item: 46);
q.insert( item: 47);
q.insert( item: 49);
q.insert( item: 55);
q.insert( item: 46);
q.insert( item: 99);
q.insert( item: 58);
q.insert( item: 59);
q.insert( item: 65);
q.insert( item: 66);
System.out.println(Arrays.toString(q.array));
System.out.println("Find 65: " + q.search( data: 65));
System.out.println("Max value got: " + q.getMax());
System.out.println(Arrays.toString(q.array));
System.out.println("Max value Extracted: " + q.extractMax());
System.out.println(Arrays.toString(q.array));
System.out.println("Max value Extracted: " + q.extractMax());
System.out.println(Arrays.toString(q.array));
```

<u>**Sample Output**</u>



```
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Queue is overflow
[45, 46, 47, 49, 55, 46, 99, 58, 59, 65]
Find 65: true
Max value got: 99
[45, 46, 47, 49, 55, 46, 99, 58, 59, 65]
Max value Extracted: 99
[45, 46, 47, 49, 55, 46, 58, 59, 65, 0]
Max value Extracted: 65
[45, 46, 47, 49, 55, 46, 58, 59, 0, 0]

Process finished with exit code 0
```

Task2 Description

Task 02: (Priority Queue using sorted array)

After completing **Task 01,** your task is to modify task 01 to implement priority queue using sorted array (as discussed in lecture 8.1). You just have to modify **insert** method to add element in ascending order and **extractMax** and **getMax** method to return last element.

In this case, running time for insert will be $O(n)$ and for extractMax/getMax will be $O(1)$.

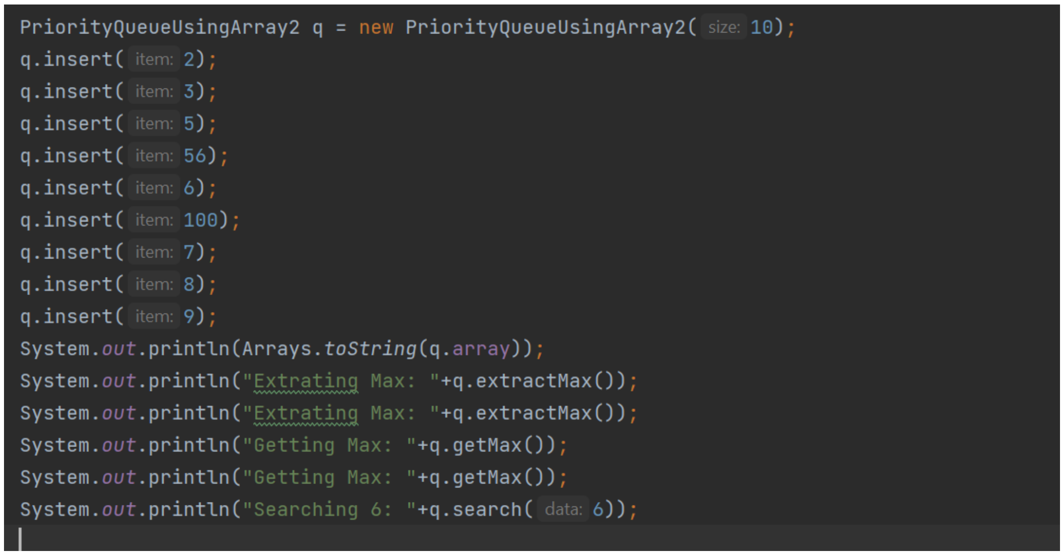
Solution:

```
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Queue;

public class PriorityQueueUsingArray2 {
    //sorted Array
    private int[] array;
    private int rear = -1;
    private int size;

      public PriorityQueueUsingArray2(int size) {
          this.size = size;
          array = new int[size];
      }

      public void insert(int item) {
          if (rear + 1 == size) {
              System.out.println("Queue is overflow");
          } else {
              if (rear == -1) {
                  array[++rear] = item;
                  return;
              }

              boolean condition = false;
              int temp = 0;

              for (int i = 0; i <= rear; i++) {


                  if (!(item > array[i]) && !condition) {
                      temp = array[i];
                      array[i] = item;
                      condition = true;
                      continue;
```

```
                }

                if (condition) {

                    int temp2 = array[i];
                    array[i] = temp;
                    temp = temp2;
                }
            }
            if (!condition) {
                array[++rear] = item;
            } else {
                array[++rear] = temp;
            }
            System.out.println("Successfully added!");
        }

    }

    public int getMax() {
        if (rear == -1) {
            System.out.println("Queue is Underflow");
            return -1;
        }
        return array[rear];

    }

    public int extractMax() {
        if (rear == -1) {
            System.out.println("Queue is Underflow");
            return -1;
        }
        int temp = array[rear];
        array[rear] = 0;
        rear--;
        return temp;
    }

    public boolean search(int data) {
        if (rear == -1) {
            System.out.println("Queue is Underflow");
            return false;
        }

        for (int i = 0; i <= rear; i++) {
            if (data == array[i]) {
                return true;
            }
        }
```

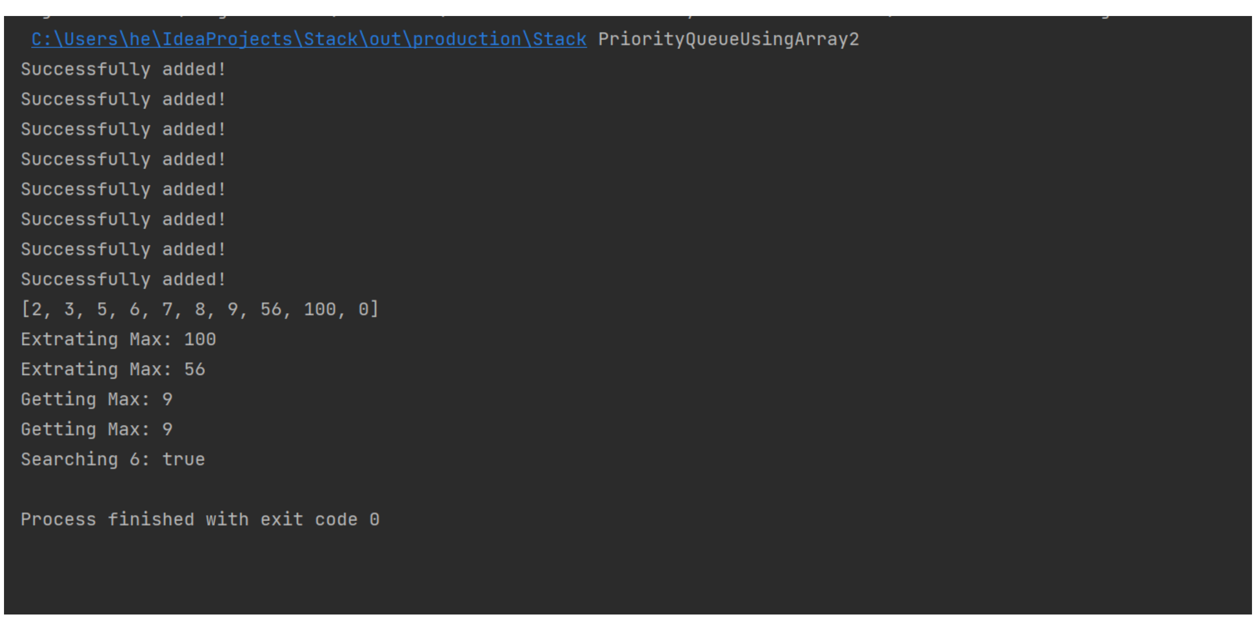
```java

        return false;


    }

    public static void main(String[] args) {

        PriorityQueueUsingArray2 q = new PriorityQueueUsingArray2(10);
        q.insert(2);
        q.insert(3);
        q.insert(5);
        q.insert(56);
        q.insert(6);
        q.insert(100);
        q.insert(7);
        q.insert(8);
        q.insert(9);
        System.out.println(Arrays.toString(q.array));
        System.out.println("Extrating Max: " + q.extractMax());
        System.out.println("Extrating Max: " + q.extractMax());
        System.out.println("Getting Max: " + q.getMax());
        System.out.println("Getting Max: " + q.getMax());
        System.out.println("Searching 6: " + q.search(6));


    }
}
```

Sample Input:



```java
PriorityQueueUsingArray2 q = new PriorityQueueUsingArray2( size: 10);
q.insert( item: 2);
q.insert( item: 3);
q.insert( item: 5);
q.insert( item: 56);
q.insert( item: 6);
q.insert( item: 100);
q.insert( item: 7);
q.insert( item: 8);
q.insert( item: 9);
System.out.println(Arrays.toString(q.array));
System.out.println("Extrating Max: "+q.extractMax());
System.out.println("Extrating Max: "+q.extractMax());
System.out.println("Getting Max: "+q.getMax());
System.out.println("Getting Max: "+q.getMax());
System.out.println("Searching 6: "+q.search( data: 6));
```

Sample Output



```
 C:\Users\he\IdeaProjects\Stack\out\production\Stack PriorityQueueUsingArray2
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
Successfully added!
[2, 3, 5, 6, 7, 8, 9, 56, 100, 0]
Extrating Max: 100
Extrating Max: 56
Getting Max: 9
Getting Max: 9
Searching 6: true

Process finished with exit code 0
```

Task3 Description

…
…
…
…

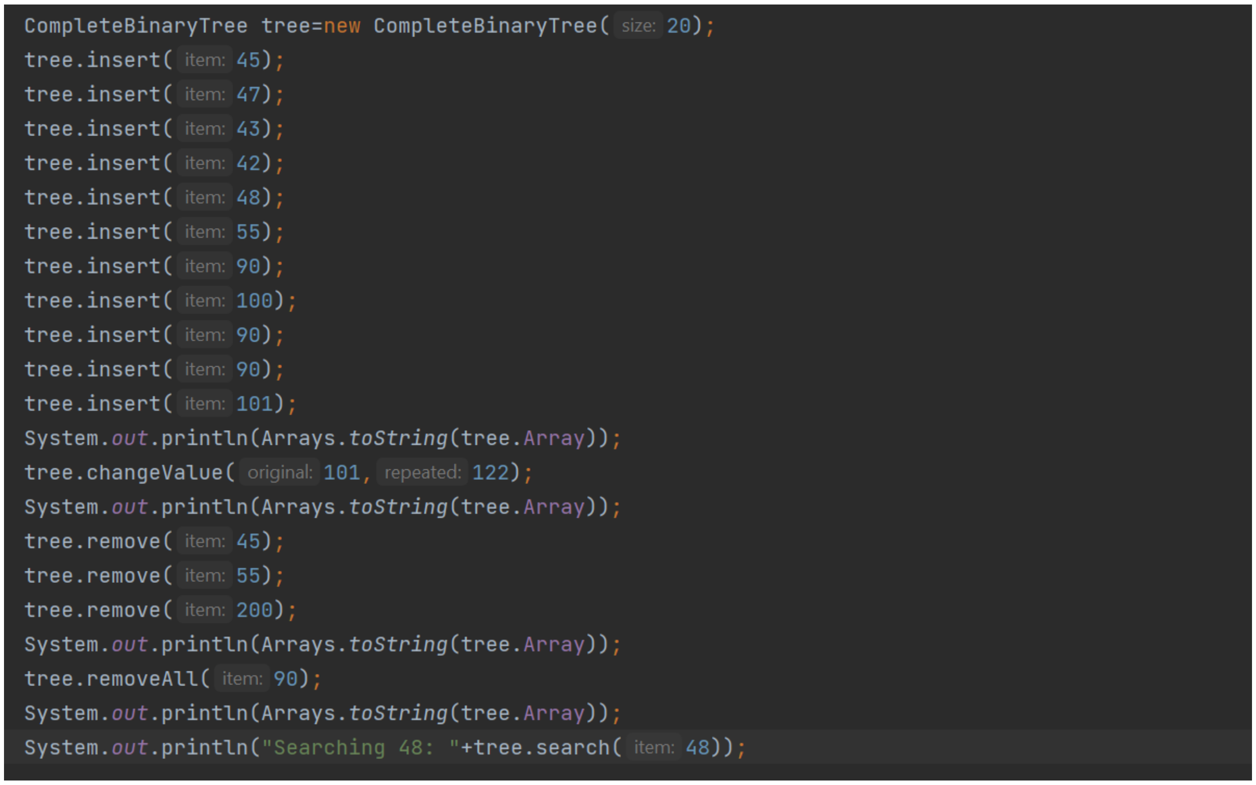
Solution:

```java
import java.util.Arrays;

public class CompleteBinaryTree {
    int[] Array;
    int size;
    int rear = -1;

    CompleteBinaryTree(int size) {
        this.size = size;
         Array = new int[size];
     }

     public void insert(int item) {
         if (rear + 1 == size) {
             System.out.println("Tree is Overflow");
             return;
         }

         Array[++rear] = item;
         System.out.println("Successfully Added");
     }

     public void remove(int item) {
         if (rear == -1) {
             System.out.println("Tree is underflow");
             return;
         }
         boolean cond = false;
         for (int i = 0; i <= rear; i++) {
             if (Array[i] == item) {
                 cond = true;
                 Array[i] = Array[rear];
                 Array[rear] = 0;
                 rear--;
                 break;
             }
         }

         if (cond)
             System.out.println("Succesfully Removed");
         else
             System.out.println("Item not found in tree");
     }

     public void removeAll(int item) {
         if (rear == -1) {
             System.out.println("Tree is underflow");
             return;
         }
         boolean cond = false;
         for (int i = 0; i <= rear; i++) {
             if (Array[i] == item) {
                 cond = true;
                 Array[i] = Array[rear];
                 Array[rear] = 0;
                 rear--;
             }
         }

         if (cond)
             System.out.println("Succesfully Removed");
         else
             System.out.println("Item not found in tree");
     }

     public void changeValue(int original, int repeated) {
         if (rear == -1) {
             System.out.println("Tree is underflow");
             return;
         }
         boolean cond = false;
         for (int i = 0; i <= rear; i++) {
             if (Array[i] == original) {
                 cond = true;
                 Array[i] = repeated;
             }
         }

         if (cond)
             System.out.println("Succesfully Changed");
         else
```

```
            System.out.println("Item not found in tree");
    }

    public boolean search(int item) {
        if (rear == -1) {
            System.out.println("Tree is underflow");
            return false;
        }
        for (int i = 0; i <= rear; i++) {
            if (Array[i] == item) {
                return true;
            }
        }

        return false;

    }

    public static void main(String[] args) {

        CompleteBinaryTree tree = new CompleteBinaryTree(20);
        tree.insert(45);
        tree.insert(47);
        tree.insert(43);
        tree.insert(42);
        tree.insert(48);
        tree.insert(55);
        tree.insert(90);
        tree.insert(100);
        tree.insert(90);
        tree.insert(90);
        tree.insert(101);
        System.out.println(Arrays.toString(tree.Array));
        tree.changeValue(101, 122);
        System.out.println(Arrays.toString(tree.Array));
        tree.remove(45);
        tree.remove(55);
        tree.remove(200);
        System.out.println(Arrays.toString(tree.Array));
        tree.removeAll(90);
        System.out.println(Arrays.toString(tree.Array));
        System.out.println("Searching 48: " + tree.search(48));


    }
```

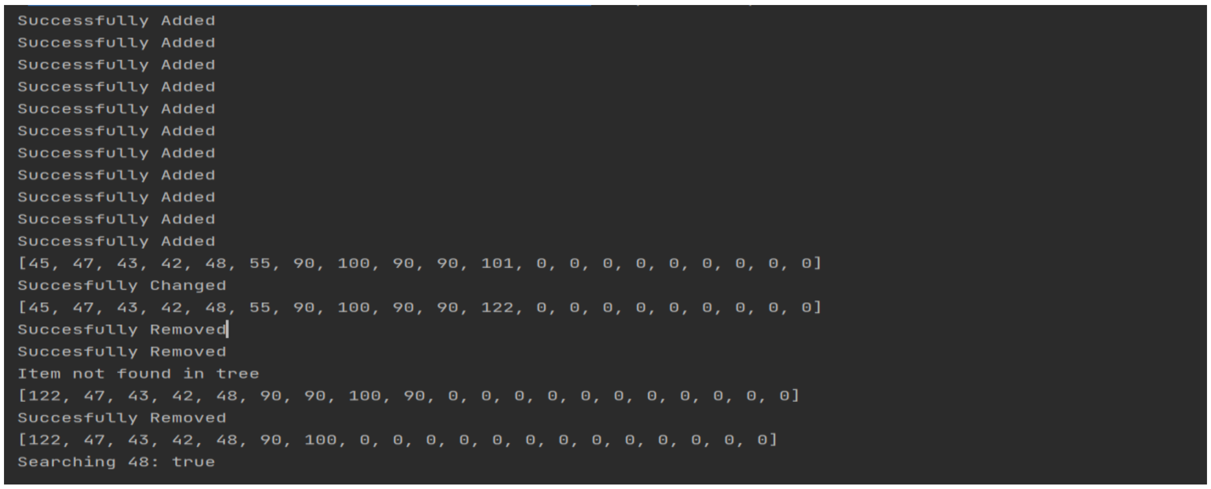
```
}
```

Sample Input:



```
CompleteBinaryTree tree=new CompleteBinaryTree( size: 20);
tree.insert( item: 45);
tree.insert( item: 47);
tree.insert( item: 43);
tree.insert( item: 42);
tree.insert( item: 48);
tree.insert( item: 55);
tree.insert( item: 90);
tree.insert( item: 100);
tree.insert( item: 90);
tree.insert( item: 90);
tree.insert( item: 101);
System.out.println(Arrays.toString(tree.Array));
tree.changeValue( original: 101, repeated: 122);
System.out.println(Arrays.toString(tree.Array));
tree.remove( item: 45);
tree.remove( item: 55);
tree.remove( item: 200);
System.out.println(Arrays.toString(tree.Array));
tree.removeAll( item: 90);
System.out.println(Arrays.toString(tree.Array));
System.out.println("Searching 48: "+tree.search( item: 48));
```

Sample Output



```
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
Successfully Added
[45, 47, 43, 42, 48, 55, 90, 100, 90, 90, 101, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Succesfully Changed
[45, 47, 43, 42, 48, 55, 90, 100, 90, 90, 122, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Succesfully Removed
Succesfully Removed
Item not found in tree
[122, 47, 43, 42, 48, 90, 90, 100, 90, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Succesfully Removed
[122, 47, 43, 42, 48, 90, 100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Searching 48: true
```