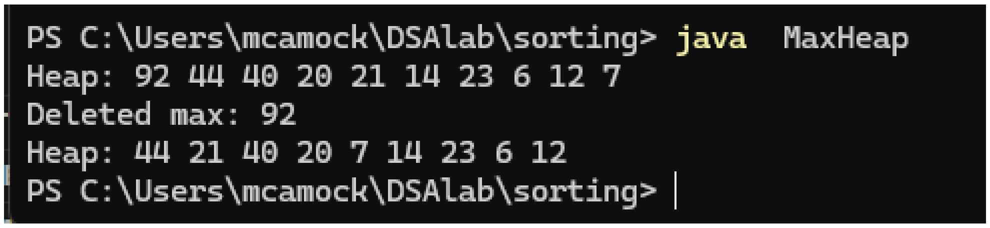
Program No:	**6**
Roll No :	**1525**
Title of Program :	
Objective :	Breadth first Traversal

SOURCE CODE:

MaxHeap.java

```
import java.util.*;

public class MaxHeap
{
  private int[] heap;
  private int size;
  private int capacity;

  // Constructor
  public MaxHeap(int capacity) {
    this.capacity = capacity;
    this.heap = new int[capacity];
    this.size = 0;
  }

  // Parent index
  private int parent(int i) {
    return (i - 1) / 2;
  }

  // Left child index
  private int leftChild(int i) {
    return (2 * i) + 1;
  }

  // Right child index
  private int rightChild(int i) {
    return (2 * i) + 2;
  }

  // Insert an element in the heap
  public void insert(int value) {
    if (size == capacity) {
      System.out.println("Heap is full");
      return;
    }
    // Insert at the end of the array
    heap[size] = value;
    int current = size;
    size++;

    // Reheap up - max heap property
    while (current > 0 && heap[current] > heap[parent(current)]) {
      swap(current, parent(current));
      current = parent(current);
    }
  }

  // Delete the root element
  public int delHeap() {
    if (size == 0) {
      System.out.println("Heap is empty");
      return -1;
    }
    int max = heap[0]; // Root element
    heap[0] = heap[size - 1]; // Move last element to root
    size--;

    reheapDown(0);
    return max;
  }

  //
  private void reheapDown(int i) {
    int largest = i;
    int left = leftChild(i);
    int right = rightChild(i);

    // Find the larger of the left and right child
    if (left < size && heap[left] > heap[largest]) {
      largest = left;
    }
    if (right < size && heap[right] > heap[largest]) {
      largest = right;
    }
    // If largest is not the root - swap and continue
    if (largest != i) {
      swap(i, largest);
      reheapDown(largest);
    }
  }

  // Swap elements
  private void swap(int i, int j) {
    int temp = heap[i];
    heap[i] = heap[j];
    heap[j] = temp;
  }

  // Display the heap
  public void display() {
    System.out.print("Heap: ");
    for (int i = 0; i < size; i++) {
      System.out.print(heap[i] + " ");
    }
    System.out.println();
  }

    public static void main(String[] args)
    {
            MaxHeap h = new MaxHeap(10);
            h.insert(23);
            h.insert(7);
            h.insert(92);
            h.insert(6);
            h.insert(12);
            h.insert(14);
            h.insert(40);
            h.insert(44);
            h.insert(20);
            h.insert(21);

            h.display(); // Display the heap

            System.out.println("Deleted max: " + h.delHeap());
            h.display();
        }
}
```

OUTPUT:



```
PS C:\Users\mcamock\DSAlab\sorting> java  MaxHeap
Heap: 92 44 40 20 21 14 23 6 12 7
Deleted max: 92
Heap: 44 21 40 20 7 14 23 6 12
PS C:\Users\mcamock\DSAlab\sorting>
```