Task 3
Implementing Heap Operations
Code a min-heap in C# with methods for insertion, deletion, and fetching the minimum element. Ensure that the heap property is maintained after each operation."

ANS:

```
package Day13;
import java.util.Arrays;
public class Task3 {
  private int[] heap;
  private int size;
  private int capacity;
  public Task3(int capacity) {
     this.capacity = capacity;
     heap = new int[capacity];
     size = 0;
  }
  public int getMin() {
     if (size == 0) {
        throw new IllegalStateException("Heap is empty.");
     }
     return heap[0];
  }
  public void insert(int value) {
     if (size == capacity) {
        throw new IllegalStateException("Heap is full.");
     }
     heap[size] = value;
     size++;
     heapifyUp(size - 1);
  }
  public int extractMin() {
     if (size == 0) {
        throw new IllegalStateException("Heap is empty.");
     }
     int min = heap[0];
     heap[0] = heap[size - 1];
     size--;
     heapifyDown(0);
```

```
    return min;
}
private void heapifyUp(int index) {
    int parent = (index - 1) / 2;
    while (index > 0 && heap[parent] > heap[index]) {
        swap(parent, index);
        index = parent;
        parent = (index - 1) / 2;
    }
}
private void heapifyDown(int index) {
    int leftChild = 2 * index + 1;
    int rightChild = 2 * index + 2;
    int smallest = index;
    if (leftChild < size && heap[leftChild] < heap[smallest]) {
        smallest = leftChild;
    }
    if (rightChild < size && heap[rightChild] < heap[smallest]) {
        smallest = rightChild;
    }
    if (smallest != index) {
        swap(smallest, index);
        heapifyDown(smallest);
    }
}
private void swap(int i, int j) {
    int temp = heap[i];
    heap[i] = heap[j];
    heap[j] = temp;
}
@Override
public String toString() {
    return Arrays.toString(Arrays.copyOf(heap, size));
}
// Test the implementation
public static void main(String[] args) {
    Task3 minHeap = new Task3(10);
```

```
        minHeap.insert(4);
        minHeap.insert(2);
        minHeap.insert(7);
        minHeap.insert(1);
        minHeap.insert(9);
        minHeap.insert(5);
        System.out.println("Minimum element: " + minHeap.getMin());
        System.out.println("Extracted minimum element: " +
minHeap.extractMin());
        System.out.println("Minimum element after extraction: " +
minHeap.getMin());
    }
}
```

OUTPUT:

```
Minimum element: 1
Extracted minimum element: 1
Minimum element after extraction: 2
```