1.
```
import java.util.concurrent.RecursiveTask;

class HeapVerifyTask extends RecursiveTask<Boolean>
{
    final int[] heap;
    final int root;

    HeapVerifyTask(int[] heap, int root)
    {
        this.heap = heap;
        this.root = root;
    }

    static int left(int i)
    {
        return 2 * i;
    }

    static int right(int i)
    {
        return 2 * i + 1;
    }

    public Boolean compute()
    {
        if (left(root) >= heap.length) {
            // Root is a leaf, so it's trivially a heap.
            return true;
        }
        else if (right(root) >= heap.length) {
            return heap[root] <= heap[left(root)];
        }
        else {
            if (heap[root] <= heap[left(root)] && heap[root] <= heap[right(root)]) {
                HeapVerifyTask left_task = new HeapVerifyTask(heap, left(root));
                HeapVerifyTask right_task = new HeapVerifyTask(heap, right(root));
                left_task.fork();
                return left_task.join() && right_task.compute();
            }
            else {
                return false;
            }
        }
    }
}
```

2. Final spanning tree: