Skew heaps:

- Skew heap = simpler variation of Leftist heap (no rank information)

- [Heap-ordering property]
  The key at each node is ≤ the keys of its children (assuming min-ordered heap)

Mergeable priority queue ADT:
- insert (x)
- removeMin( )  [or removeMax( )]
- merge (heap1, heap2) ⇒ return a new skew heap that has all the keys of heap1 and heap2

merge (heap1, heap2) {
  merge the rightmost paths of heap1 and heap2;
  for each node encountered on the new rightmost path swap it with its left sibling;  // no rank info
}

Next we illustrate the merge operation.
Heap1: 8

Heap2: 5

Heap1’s rightmost path: 8, 13, 18  Heap2’s rightmost path: 5, 17

Begin by merging these rightmost paths: 5, 8, 13, 17, 18
Next swap nodes along the rightmost path with their siblings
(So nodes on the rightmost path move to the leftmost path)
Merge is often implemented using one downward pass:

```
Heap merge (heap1, heap2) {
    Node root = merge (heap1.root, heap2.root);
    return new Heap (root);
}

Node merge (Node p1, Node p2) {
    if (p1==null)
        return p2;
    else if (p2==null)
        return p1;
    else if (p1.key < p2.key) {
        Node temp = merge (p1.right, p2);
        return new Node (p1.key, temp, p1.left);
    }
    else /* (p2.key <= p1.key) */ {
        Node temp = merge (p1, p2.right);
        return new Node (p2.key, temp, p2.left);
    }
}
```
Next we consider the insert and removeMin operations.

\[
\text{insert (} x \text{)} \{ \\
\quad \text{create NewHeap with one key } x; \\
\quad \text{merge (} \text{CurrentHeap}, \text{NewHeap}); \\
\}
\]

\[
\text{removeMin( ) } \{ \\
\quad \text{temp = root.key; } \\
\quad \text{merge (} \text{root.left}, \text{root.right}); \\
\quad \text{return temp; } \\
\}
\]

Beginning with an empty min-ordered skew heap, any sequence of \( m \) operations (inserts, removeMins, merges) runs in \( O(m \lg n) \) total time, where \( n = \) max number of keys.

So average time per operation is \( O(m \lg n / m) = O(\lg n) \).

Finally we illustrate the insert and removeMin operations.
Insert 5:
RemoveMin:

```
4
   10
  /  \
13   18
    /  \
  20   12
     /    \
25
```

Merge:

```
10
 /  \
13   18
    /  \
  20   12
     /    \
25
```

```
7
 /  \
20   10
   /  \
13   12
   /    \
25   18
```

```
7
 /  \
10   20
   /  \
12   13
   /    \
18   25
```