Bill Davis
605.421

1. 19.1-3 Binary Representation of Binomial Heaps
   There are $\binom{n}{k}$ binary $k-$strings containing exactly j 1's. And since there are also $\binom{n}{k}$ nodes per level of a binomial tree, there must be k ones per node per level of a tree when counting the lowest node as 0.

   Since binomial trees have defined sizes, ie they either have 1,4,8,16 ... nodes, $2^0, 2^1, 2^2, 2^3...$, a node of degree k will have subtrees of size $2^0 + 2^1 + 2^2 + ...2^{k-1} = 2^{k-1}$, therefore it will have k 1's in it's binary representation.

2. 19.2-5 Binomial-Heap-Minimum
   Binomial-Heap-Minimum may not work as coded. If all of the nodes in the heap have value ∞, then Binomial-Heap-Minimum will return NULL, even though there are elements in the Heap.

   It can be altered to account for this by,

   y $\leftarrow$ *head*[H] // This sets a default return value
   x $\leftarrow$ *head*[H]
   min $\leftarrow \infty$
   while x $\neq$ NILL
       do if *key*[x] $<$ *min*
           do if *key*[x]$<$ *min*
               then *min* $\leftarrow$ *key*[x]
                   $y \leftarrow x$
           $x \leftarrow$ *sibling*[x]
   return y

   This way if there is an element $< \infty$ we will select it, otherwise we'll return the first value.

3. 19.2-6 Binomial-Heap-Delete
   We can alter Binomial-Heap-Delete to work in the case where there is no representation for $-\infty$.

   Binomial-Heap-Delete(H,x)

y = Binomial-Heap-Extract-Min(H)
Binomial-Heap-Decrease-Key(H,x, y-1)
Binomail-Heap-Extract-Min(H)
Binomial-Insert(y)

In this manner they key x always has the lowest value. It simply requires two deletes and an insert, all of which can be done in $\mathbf{O}(\lg n)$

4. Minimum spanning tree with binomal heaps
A binomial heap can be used to manage both the edge and vertex list. For the edge list the operations are straightforward. Extracting the minimum-weight edge is simply Binomial-Heap-Extract-Min, and $E_i \leftarrow E_i \bigcup E_j$ is the Union operation.

   Vertex operations are slightly more complicated. Once we extract an edge from the edge list we need to determine which heaps the endpoints belong in. This requires a new operation to search the heap looking for elements. For example, once we extract the minimum-weight edge (u, v) from E_i, we have to determine which V_i and V_j, u and v belong in. After this we can reuse the union operation to merge V_i and V_j if neccessary.

   The runtime of this algorithm should be E lg(E), since we require approx. E Extract-Min operations which happen in E time.