Fibonacci Heap Extract

The *findMin* and *extractMin* functions:

```
function findMin(h)    // h is the heap
    {
    return h.min.value;
    }

function extractMin(h) // h is the heap
    {
    var result = h.min.value;
    var c = h.min.childlist;
    h.size = h.size - 1;
    h.rootlist = consolidate(h.size,append(remove(h.min),c));
    h.min = h.rootlist;
    return result;
    }

function consolidate(n,r)
    {
    if (r == null) return; //nothing to consolidate

    var D = makeDegreeArray(log2(n)+1); //slots of D are initialized to null
    var spot = r;

    r.prev.next = null; //break the circle

    while (spot != null)
        {
        var d = degree(spot); //degree is the number of children
        var next = spot.next; //save the next spot since spot may change

        //link until an open slot in the degree array is found
        while (D[d] != null)
            {
            spot = link(spot,D[d]);
            D[d] = null;
            ++d;
            }
        //store the subheap in the degree array
        D[d] = spot;
        //move to the next spot in the root list
        spot = next;
        }

    //convert the degree array into a circular, doubly-linked list
    return rootify(D);
    }
```

The *rootify* function walks the degree array, inserting the non-null entries into a circular, doubly-linked list. It also keeps track of the minimum, returning a pointer to that minimum node when done.

The *link* function adds the subheap with the larger root value into the child list of the other subheap. It also increments the degree of the receiving subheap.

The *append* and *remove* functions are standard circular, doubly-linked list routines. The *append* function returns a list made from the two given lists. The *remove* function removes the given node from its list, returning the modified list (which node in the list that is returned is unimportant). Both functions destroy the original lists.

Next: *Combining two Fibonacci heaps into one.*