Homework Assignment #4
Due: June 16, 2023 at 10:00 p.m.

1. We discussed in class (and it is given in detail in the textbook) to perform a UNION on two binomial heaps. It merges the two root lists and then makes a pass through the merged list to link up pairs of roots of the same degree. This is a little wasteful when the two heaps are of very different sizes; in some cases we could stop the second pass early if we can determine that no further more roots need to be linked.

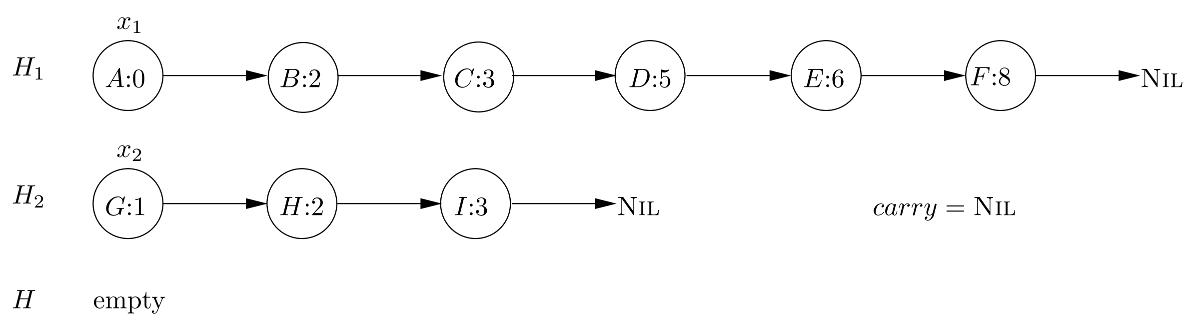
Here is a different, optimized implementation of the UNION operation that does stop early. It just makes a single pass to merge the root lists and simultaneously links pairs of roots of the same degree.

```
function UNION(H1, H2)
    x1 ← head(H1)                  ▷ first root in H1
    x2 ← head(H2)                  ▷ first root in H2
    carry ← NIL
    H ← new empty binomial heap
    loop
        exit when at most one of x1, x2, carry is non-NIL
        let d be the smallest degree among the roots x1, x2, carry that are non-NIL
        if x1, x2 and carry are all non-NIL and all three have degree d then
            append carry to the end of H's root list
            let carry be the root x1 or x2 with smallest priority and other be the other one
            x1 ← x1.next               ▷ advance to next root of H1
            x2 ← x2.next               ▷ advance to next root of H2
            carry.next ← NIL           ▷ detach roots carry and other from root lists of H1, H2
            other.next ← NIL
            make other the first child of carry
        else if two of x1, x2, carry are non-NIL and have degree d then
            let carry be the one of these two with smallest priority and other be the other one
            if x1 = carry or x1 = other then x1 ← x1.next ▷ advance to next root of H1
            end if
            if x2 = carry or x2 = other then x2 ← x2.next ▷ advance to next root of H2
            end if
            carry.next ← NIL           ▷ detach roots carry and other from root lists of H1, H2
            other.next ← NIL
            make other the first child of carry
        else                           ▷ all of the non-NIL nodes among x1, x2, carry have different degrees
            let next be the root among x1, x2, carry that has degree d
            if next = x1 then
                x1 ← x1.next           ▷ advance to next root of H1
                next.next ← NIL        ▷ detach next from H1's root list
            else if next = x2 then
                x2 ← x2.next           ▷ advance to next root of H2
                next.next ← NIL        ▷ detach next from H2's root list
            end if
            append next to H's root list
            carry ← NIL
        end if
    end loop
    if one of x1, x2, carry is non-NIL then
        append that root to the end of H's root list
    end if
    return H
end function
```

If the two heaps have sizes n_1 and n_2 the UNION of the heaps is done analogously to doing binary addition of n_1 and n_2. In particular, the root stored in *carry* behaves like the carry bit in the binary addition.

If x_1 is non-nil when the loop exits, line 40 appends the rest of H_1's root list to the end of H's root list. Similarly, if x_2 is non-nil when the loop exits, line 40 appends the rest of H_2's root list to H's root list. This saves some time because the algorithm does not have to traverse the rest of those lists.

[3] **(a)** Suppose H_1 is a binomial heap containing 365 items and H_2 is a binomial heap containing 14 items. If we do a UNION(H_1, H_2), the state of the data structure at the start of the first iteration of the loop is shown below. We only show roots of the binomial trees, and each root is labelled with a name and the node's degree.



Draw similar diagrams showing showing these 9 nodes after each iteration of the loop. Indicate which roots the pointers x_1, x_2 and *carry* point to at the end of each iteration. Then, draw a similar diagram showing the heap H that is returned by the UNION.

[1] **(b)** How does each iteration of the loop in UNION where the test on either line 9 or 17 is true change the total number of tree roots?

[2] **(c)** Consider a UNION on two heaps containing n_1 and n_2 items. Show that there are at most $1 + \lceil \log(\min(n_1, n_2)) \rceil$ iterations of the loop where the tests on line 9 and 17 are both false.

[1] **(d)** Show that $\log_2(n_1 + n_2) \leq 1 + \log_2(\max(n_1, n_2))$.

[4] **(e)** As discussed in class, an INSERT(x) into H_1 can be performed by creating a heap H_2 containing a single node and then performing a UNION(H_1, H_2). A MAKEHEAP operation creates a new empty heap. Consider a sequence of operations that consists only of MAKEHEAP, INSERT and UNION operations. Assume that initially there are no heaps. Do an amortized analysis of this sequence.

Hint: as mentioned in class, the INSERTS can be analyzed by storing one unit of potential (or one dollar, if you prefer the accounting method) for each root of each heap. To handle UNION operations, add $\log n$ units of potential (or dollars) for each heap containing $n > 0$ elements and use parts (b)-(d).