**Assignment 6**

**Level 1:  
1. (a) Given a binary tree *T* containing n keys and a key *k*, design a recursive pseudo-code  
algorithm FindSmallerKeys(T, k) that returns a Sequence of keys in *T* that are less than  
or equal to key *k*. For example, given the heap at the bottom slide 34 of the notes and  
query key k=7, the algorithm should return 7, 5, 4, 6. Note that the keys do not need to be  
reported in any particular order.**

Algorithm findSmallerKeys(T,k)

if T.isEmpty() then

return Error message

S:= [ ]

findSmallerHelper(T,k,T.root(),S)

return S

Algorithm findSmallerHelper(T,k,v,S)

if T.isExternal(v) then

return

else

if v.element() <=k then

S.insertLast(v)

findSmallerHelper(T,k,T.leftChild(v),S)

findSmallerHelper(T,k,T.rightChild(v),S)

return S

**(b) Implement your pseudo-code algorithm in JavaScript.**

function findSmallerKeys(T, key) {

    let seq = new Seq.Sequence();

    findSmallerHelper(T, key, T.root(), seq);

    console.log("findSmallerKeys key=" + key + " " + seq.toString());

    return seq;

}

function findSmallerHelper(T, key, p, seq) {

    // Your code goes here

    if(T.isExternal(p)){

        return ;

    } else{

        if(p.element() <= key){

            seq.insertLast(p);

        }

        findSmallerHelper(T,key,T.leftChild(p),seq);

        findSmallerHelper(T,key,T.rightChild(p),seq);

    }

return seq;

}

**2. Suppose a binary tree T is implemented using an array S, as described in the notes. If n  
items are stored in S in sorted order, starting with index 1, is the tree T a heap? Justify  
your answer.**

The array is a min-heap in which all the elements are sorted and the value of parent is lesser than the value of its children.

**R-2-18 Draw an example of a heap whose keys are all the odd numbers from 1 to 49 (with no  
duplicates), such that the insertion of an item with key 32 would cause up-heap bubbling to  
proceed all the way up to a child of the root (replacing that child key of the root with 32).**

**Level 2:  
1. This is a refinement of question 1 above but it takes advantage of the fact that *T* is  
a Heap for improving efficiency.  
C-2.32 Let *T* be a heap storing n keys. Design an efficient recursive pseudo-code algorithm for  
reporting all the keys in T that are smaller than or equal to a given query key x (note that x is not  
necessarily in T). Ideally, your algorithm should run in O(k) time, where k is the number of keys  
reported. Hint: Stop searching beyond a node when the key is greater than x.**

Algorithm findSmallerKeys(T,k)

if T.isEmpty() then

return Error message

S:= [ ]

findSmallerHelper(T,k,T.root(),S)

return S

Algorithm findSmallerHelper(T,k,v,S)

if v.element() >=k then

return

else

S.insertLast(v)

findSmallerHelper(T,k,T.leftChild(v),S)

findSmallerHelper(T,k,T.rightChild(v),S)

return S