COSCI 21a, Assignment 5

Solution Hints

1. Describe in English and give pseudo-code for a recursive algorithm that returns the last vertex in a singly linked list $L$ that contains data $d$ (or nil if $d$ is not in $L$); that is, if there is more than one copy of $d$ in $L$, then it returns the last one encountered when traversing the list from the first to the last item. For this problem, assume that a singly linked list does not have a header and is represented by a pointer to its first element.

We give a recursive algorithm $LastSearch$ to search forward from a vertex, and call it on the first vertex of $L$. $LastSearch$ returns nil if it is passed a nil pointer; otherwise, it first calls itself recursively on the tail of the list to see if $d$ is there, and if not it checks if the head of the list contains $d$, and if not returns nil.

```plaintext
function LastSearch(d, p)
    if p = nil then return nil
    else begin
        q = LastSearch(d, NEXT(p))
        if q ≠ nil then return q
        else if DATA(p) = d then return p
        else return nil
    end
end

LastSearch(FIRST(L))
```
2. We have seen that both the recursive and iterative versions of the Towers of Hanoi Problem use $O(n)$ space. Compare the constants that would be expected in practice, charging one unit of space to store an integer (assume that a return address can be stored as an integer). Clearly state any other assumptions you are making.

Tower uses only $O(1)$ explicitly declared space, but "behind the scenes", the stack used to implement the recursion goes up and down, but never to a depth greater than $n$. Each frame has the variables $n, x, y, z$ and the return address. Hence, $5n + O(1)$ units of space are used (the $O(1)$ covers any local temporary variables when actually implementing this code at a low level).

Now recall the iterative algorithm:

```plaintext
if $n$ is odd then $d := clockwise$ else $d := counterclockwise$
repeat
  Move the smallest ring one post in direction $d$.
  Make the only legal move that does not involve the smallest ring.
until all rings are on the same post
```

The straightforward implementation declares three stacks of size $n$ to represent the three posts, using $3n + O(1)$ units of space (the $O(1)$ covers the variables $n, d$, and any other indexing variables, etc. when actually implementing this pseudo code). So already, we can answer that the iterative algorithm uses less space.

For fun, we can see if we can do better. The total number of stack frames used at any point in time is $n$, and so the possibility exists to reclaim some of this $2n$ "wasted" space. One simple way to reclaim $n$ of it is to have two of the stacks share the same array of size $n$, where the two stacks grow from opposite ends of the array, thus bringing the space down to $2n + O(1)$. So now we are now already more than a factor of two better than the recursive algorithm.

Another approach is to store for each disk, the index of the post it is currently on (from which the index of the top disk on each post can be deduced at each step), for a total of $n + O(1)$ space (but using more time).
3. Starting with an empty 2-3 tree, using the algorithm presented in the course notes, draw the 2-3 tree that results from inserting the following strings in this order (using the usual English alphabetical ordering):

\[ \text{horse, cow, pig, seal, rat, dog} \]

\[
\begin{array}{c}
\text{horse, rat} \\
\text{cow, dog} \\
\text{pig} \\
\text{seal}
\end{array}
\]

The draw the 2-3 tree that results from continuing and additionally inserting these strings:

\[ \text{goat, elephant, fish, rooster, zebra, roach, cat} \]

\[
\begin{array}{c}
\text{dog, fish} \\
\text{cat, cow} \\
\text{elephant} \\
\text{goat} \\
\text{pig} \\
\text{roach, rooster} \\
\text{zebra}
\end{array}
\]

And finally, draw the 2-3 tree that results from continuing and additionally inserting these strings:

\[ \text{hen, llama, aardvark, hog, donkey, rhino, hippo, tiger} \]

\[
\begin{array}{c}
\text{dog} \\
\text{cat} \\
\text{aardvark} \\
\text{cow} \\
\text{donkey} \\
\text{elephant} \\
\text{fish, hen} \\
\text{rat} \\
\text{seal} \\
\text{llama, pig} \\
\text{rhino} \\
\text{rooster} \\
\text{tiger, zebra}
\end{array}
\]
4. Draw a red-black tree equivalent to the 2-3 tree you constructed for Problem 3.

In this figure, thick line circles indicate a black vertex: