Due: Wednesday, October 28, 11AM

This assignment is worth 15% of your final grade

1. **NOTE: I was not careful in defining a binary search tree.** What I had in mind was a *strict* binary subtree where every internal node has degree 2 (i.e. two children) which is defined only when the number of nodes n is odd. For n even, so as to have a definition that captures this idea of being strictly binary, let's say that a strict binary tree satsifes on an odd number of entries satisfies the condition that every internal node has degree 2 except for one internal node whose single child is a leaf. (This is not standard as strict binary search tree usually only applies to n odd but lets be generous in our definition of strict.)

   The definition I gave for *balanced* applies to strict binary search trees as I have defined them above. Namely, we will say that a strict binary search tree is balanced if there exists an ℓ such that the depth (level) of every leaf is either ℓ or $\ell + 1$.

   Consider storing the following names in a binary search tree:
   $\{allan, diane, margo, norman, philip, susan, ted, valerie\}$.

   - Draw a balanced strict binary search tree for these names. For a balanced strict binary search tree on n nodes, what is the maximum number (as a function of n) of accesses to the tree to determine whether or not a name is in the tree? Be as precise as possible.
   - Draw the most unbalanced strict binary tree possible. Is there a unique way to do this?.

2. In practice we use an explicit deterministic (i.e. no randomization) hash function such as the one mentioned in class: $h(ID) = a \cdot ID + b (mod\ p)(mod\ m)$ to map some number of possible IDs into m storage locations. Here p is a large prime with $p > m$ and a and b are fixed numbers (chosed randomly) in $\{1, 2, ...p - 1\}$. Suppose we could have a perfect hash function as in a balls and bins experiment. Formally, $h(ID) = a \in \{0, 1, \ldots m - 1\}$ where we randomly choose a each time we want to insert an item into the hash table. Why do we still want to use some explicit deterministic hash function? Does it make sense to use such a hash function if $n < m$ where n is the actual number of entries in the hash table?

3. Suppose that I invent a new programming language such that every Algorithm written in this langauge must halt. Can such a programming language allow us tocompute all Turing machine computable functions? Explain your answer.Note (just for interest): Such languages do exist. Without spelling out the details, one would use FOR loops and not While loop.

4. We want to consider two different types a search experiences, for each of which you will create a record of your search experience as will be explained. In both cases, do not search for something that would be considered personal.

(a) Create an "information need" for which you believe that the desired information exists in a single document but where you are not sure how to create an appropriate query. Create an initial query and detail your search experience. Namely, record that query, the time you issued the query, the titles of the 10 highest ranked documents returned, and whether or not you obtained the desired information. If you obtained the desired information then indicate the rank of the relevant document and try one more information need. If you didnt find the desired information, refine your query and try searching again using your refined query. Try explaining as best you can how your thinking led to the refined query. Indicate if you were successful or not in your refined search and if successful what was the rank of the relevant document. Wait one daa yand repeat the same search for your information need using the same initial query. Did you obtain the same set of the top 10 ranked documents?

(b) For this part of the question consider searching for an information need that you beieve does not exist in a single document but does exist by combining inforamtion from two or three documents. Once again, record you search experience as above. If your initial search returns a single document (in the top ten) then try one more information need. If the desired information exists in two or three of the top ranked documents, then indicate that. Otherwise refine your query to see if the desired information can be found amongst the top ranked documents from both the initial and refined queries. In this latter case, explain as best you can, your thinking that led to the refined query.