1) (10 pts) ANL (Algorithm Analysis)

There is a very long corridor of rooms, labeled 1 through n, from left to right. It is reputed that in the very last room, room *n*, there is the Treasure of the Golden Knight. Unfortunately, you don't know what *n* is equal to. Whenever you are in a particular room, you are allowed to ask questions of the form, "Is there a room 2^k slots to the right of my current location?", where k is a non-negative integer. For a fee, Knightro, an omnipresent, omnipotent, omniscient knight, will answer your question correctly, with either "yes" or "no." After you ask 1 or more questions from a single room, Knightro will move you, for free, to any of the rooms you asked a question about for which he replied "yes." Your goal is to get to room n by asking as few questions as possible, to reduce the fee that you pay Knightro. Devise a strategy to find the value of n and clearly outline this strategy. How many questions, in terms of n, will your strategy use, in the worst case? Answer, with proof, this last question with a Big-Oh bound in terms of n. **(Note: Any strategy that works will be given some credit. The amount of credit given will be based on how efficient your strategy is, in relation to the intended solution.)**

One strategy that is fairly efficient, is as follows:

1. When in room 1, ask the questions for each value of k successively, until receiving the first no answer. Thus, if there is a room 2^m rooms to the right, but NOT 2^{m+1} rooms to the right, stop asking questions and ask to be moved to the room 2^m rooms to the right, room $1+ 2^m$.

2. If there was no "yes" response at all during step 1, then the answer is n = 1. If this isn't the case, go onto the next steps.

3. Create a variable called cJump and set it equal to m.
4. Let cur equal the current room number you are in.
5. While cJump is greater than equal to 0, do the following:

    a. Ask the question, "Can I move to the right by $2^{curJump}$ number of rooms?
    b. While curJump is non-negative and the answer to question in step a is no, subtract 1 from curJump.
    c. If curJump is non-negative, update cur by adding $2^{curJump}$ to it (asking Knightro to move you)

Step one takes O(log n) steps, since we know that $2^m <= n$ and $2^{m+1} > n$. Solving the former inequality shows that $m <= \log_2 n$, and we ask m+1 questions, we've asked O(log n) questions in this step.

Steps 2, 3 and 4 take O(1) time. Even though the loop for #5 has a loop in it, since curJump never gets incremented, the total number of times Steps 5a and 5b run is m+1. (It's always guaranteed to decrement once every time it runs due to addition with powers of two.) Thus, the total run time of step #5 is O(log n) as well.

Adding, we get a run-time of O(log n).

Grading: 10 pts max O(lg n) strategy, 9 pts max O(lg^2n) strategy, 4 pts max O(n) strategy. Within a strategy, award points for the explanation and run-time proof as follows (7/3, 6/3, 2/2). If there is a run time better than O(n) but not as good as O(lg^2n), give something in between 4 and 9 points.