Questions:

1. What is the design value in writing an ADT?
2. When assessing complexity, what do we measure (or, what do we not measure)?
3. Does an upper bound for a function apply in all cases? Explain.
4. Why might it be important to have different measure of complexity (e.g. upper bounds, lower bounds, …)?
5. Under what conditions might you not be concerned about upper and lower bounds?
6. The lecture notes state, "Sometimes the 'worst' algorithm is the best choice." What does this mean, and how might it apply in specific situation?
7. How important is an understanding of space complexity in an era of cheap memory?

Answers:

1. Writing ADT is often crucial in designing data structures. Dividing into two segments often helps in specifying how to design the ADT. It is important to think as a designer, and consider how the user will interact with the ADT. Also important is to think as an engineer, and consider how all the pieces will fit together behind the scenes. Writing out ADT helps in this thought process
2. When measuring complexity of an algorithm using big O notation, we measure the dominant term. For polynomial functions, this is the term with the highest exponent. For big O notation, we do not measure non-dominant terms (terms with exponents less than the dominant term).
3. The upper bound of a function does not necessarily apply in all cases. The upper bound of a function does not necessarily bound the function for small data sets, just for values greater than beta. Therefore, the upper bound of a function may not apply for small data sets.
4. Just having an upper bound can tell you only the upper bound of a problem, which can be misleading. Having a lower bound as well with a different constant can give more exact approximations for the runtime of the algorithm.
5. With small data sets you may not be concerned with upper and lower bounds. Upper and lower bounds only apply at values greater than beta. So for values lower than beta, the upper and lower bounds do not apply, and can even be misleading.
6. The 'worst' function is defined as an estimate of performance based on big O notation. Since big O notation is just an estimate, it cannot predict the exact runtime of an algorithm. There are times where the estimation can be misleading, and in cases such as these, the 'worst' algorithm may wind up being the best choice.
7. Understanding space complexity is sometimes important, but we are usually more limited by the time complexity of a function. I believe a terabyte of memory costs about $40, therefore it is pretty cheap to store more data, but it seems to be more valuable