Make sure you study and understand:

1) Basic recurrence relations for arithmetic and geometric problem reduction and their solutions. Only need to study the types of problems we have covered in class so far: simple linear homogeneous recurrence relations (where sub-problems are sizes n-i) and recurrence relations that arise in divide and conquer algorithms (where sub-problems are sizes n/b).

2) General method of creating recursive algorithms for solving problems: the identification of the base cases, problem decomposition and solution construction components of a solution. Also demonstrate the ability to apply these steps for creating a recursive solution to new (but similar) problems.

3) Exactly why and how the knapsack solutions we developed work to solve the problem. Can you adapt the solution to solve other similar problems for recursive, caching and DP approaches? How does the linear space DP method work to save space? How much longer does it take? Could you adapt the linear space method and apply it to a different problem?

4) The steps needed to be taken when formulating a dynamic programming solution from a recursive solution and be able to apply those steps to solve a given recursive algorithm. The steps needed to change a recursive solution into a caching solution. What caches can be used? When is caching a good idea?

5) The methods and analysis of empirical timing studies of algorithms. How is data gathered and plotted? What is the relevance and advantages of using log/log plots or log/linear plots rather than linear/linear plots?

6) How each of the following divide and conquer algorithms work and be able to explain how their run time is influenced by the organization of the data given to them: quick sort, merge sort, k-rank value, and polynomial multiplication.