ECS 60 Previous Final ADT Essay Questions

1. ADT design (80 points) You've been assigned to design an assembly line for an automobile manufacturing plant. An assembly line is a linear set of stations. At each station, people begin and complete specific task(s). For each task your will program will receive: 1) The time it takes to complete the task, and 2) the previous task(s) that the current task is directly dependent upon being completed before it can start. There are T tasks, and a total of D dependencies for all of the tasks.

   a) (20 points) If you assume that only one task can be done at each station, how would your program determine an ordering of the tasks that will take the minimum time? In terms of D and T, what is the big-O of your solution?

   b) (60 points) If you assume that up to two tasks can be done at each station simultaneously, how would your program determine the best ordering and station of the tasks that will take the minimum time? In terms of D and T, what is the big-O of your solution?

2. ADT design (70 points) For this problem you will address two sides to the railroad business---financial and logistical.

   a) (50 points) Many railroad cars are owned by companies other than the railroad hauling them. The railroads have a contract with car owners to assure that the cars must earn at least a certain minimum amount each month. If a privately owned car is underused, a railroad must pay the owner a penalty that is geometric to underuse, e.g., the penalty for a car that misses its target by $10 might only be $5, but the penalty for a car that misses its target by $100 might be $500. Thus, the railroads must make sure that the most underused cars at a desired location are assigned work first. Therefore, they have two operations: 1) Updating the earnings of a car, based on its ID, when a cargo is delivered to a location; and 2) selecting unused cars for use based on their earnings and location. Describe and justify your choices of data structure(s) that would minimize the time used by the system. Be sure to describe how the two operations would work, and the big-O(s) involved.

   b) (20 points) A railroad dispatcher must determine the best way to route rail traffic so that the maximum amount of cargo can be transported. Each rail connection between cities can only handle a certain number of cars a day. The dispatcher has a list of all the connections available to his/her railroad and the number of cars each can carry. The New York dispatcher often has to determine the best way to send hundreds of cars from New York to Los Angeles on any given day. No single route can handle all of the traffic. Provide a method for the dispatcher to determine the best routes for the New York cars so that as many cars as possible reach Los Angeles without becoming stuck somewhere. Be sure to explain how your method would be used with this particular problem. We are not looking for a data structure, nor big-O.

3. ADT design (78 points) There are now programs that will give you street directions to get from any city in the United States to any other city in the United States. For this program, assume that cities are completely separated from each other, and are only connected by freeways. You are to describe <u>all</u> of the algorithms and data structures necessary to implement such a program efficiently given the following specifications. Remember to mention big-O wherever appropriate.

   1. The continental United States has 20,000 connected cities (**C**), and 25,000 freeways (**F**).
      1.1. Each city has a name <char [44] >, state <char [2]>, and unique city ID <int>, for a total of 50 bytes.
      1.2. Freeways.
         1.2.1. Specified by its name <char [4] >, city1 ID <int>, city2 ID <int>, length in feet <int>, and a unique road ID <int>, for a total of 20 bytes.
      1.3. Intersections
         1.3.1. There is one intersection in each city. Thus, there are as many intersections as cities, **C.**
         1.3.2. Specified as a list of the road IDs of the Freeway(s) that meet at one physical location, and the ID of the city in which the intersection occurs <int>. No more than nine Freeways can intersect at one physical location, so an intersection can be no more than 40 bytes.
   2. As input your program is given the city, and state of both the origin and destination.
   3. Your program should provide an ordered list of the IDs of the Freeway(s) that is the shortest (in feet) route from the origin to destination.
   4. Do not worry about insertions, deletions, or updates.
   5. Your program can use 10 Megs of RAM.