1. Recall the class scheduling problem described in lecture. We are given two arrays $S[1..n]$ and $F[1..n]$, where $S[i] < F[i]$ for each i, representing the start and finish times of n classes. Your goal is to find the largest number of classes you can take without ever taking two classes simultaneously. We showed in class that the following greedy algorithm constructs an optimal schedule:

   > Choose the course that *ends first*, discard all conflicting classes, and recurse.

   But this is not the only greedy strategy we could have tried. For each of the following alternative greedy algorithms, either prove that the algorithm always constructs an optimal schedule, or describe a small input example for which the algorithm does not produce an optimal schedule. Assume that all algorithms break ties arbitrarily (that is, in a manner that is completely out of your control). ***Exactly three of these greedy strategies actually work.***

   - Choose the course x that *ends last*, discard classes that conflict with x, and recurse.
   - Choose the course x that *starts first*, discard all classes that conflict with x, and recurse.
   - Choose the course x that *starts last*, discard all classes that conflict with x, and recurse.
   - Choose the course x with *shortest duration*, discard all classes that conflict with x, and recurse.
   - Choose a course x that *conflicts with the fewest other courses*, discard all classes that conflict with x, and recurse.
   - If no classes conflict, choose them all. Otherwise, discard the course with *longest duration* and recurse.
   - If no classes conflict, choose them all. Otherwise, discard a course that *conflicts with the most other courses* and recurse.
   - Let x be the class with the *earliest start time*, and let y be the class with the *second earliest start time*.
     - If x and y are disjoint, choose x and recurse on everything but x.
     - If x completely contains y, discard x and recurse.
     - Otherwise, discard y and recurse.
   - If any course x completely contains another course, discard x and recurse. Otherwise, choose the course y that *ends last*, discard all classes that conflict with y, and recurse.

2. A party of n people have come to dine at a fancy restaurant and each person has ordered a different item from the menu. Let $D_1, D_2, \ldots, D_n$ be the items ordered by the diners. Since this is a fancy place, each item is prepared in a two-stage process. First, the head chef (there is only one head chef) spends a few minutes on each item to take care of the essential aspects and then hands it over to one of the many sous-chefs to finish off. Assume that there are essentially an unlimited number of sous-chefs who can work in parallel on the items once the head chef is done. Each item D_i takes h_i units of time for the head chef followed by s_i units of time for the sous-chef (the sous-chefs are all identical). The diners want all their items to be served at the same time which means that the last item

to be finished defines the time when they can be served. The goal of the restaurant is to serve the diners as early as possible. Consider the following greedy algorithms that order the items according to different criteria. For each of them either describe a counter example that shows that the order does not yield an optimum solution or give a proof that the ordering yields an optimum solution for all instances.

- Order the items in increasing order of $h_i + s_i$.
- Order the items in decreasing order of $h_i + s_i$.
- Order the items in increasing order of h_i.
- Order the items in decreasing order of h_i.
- Order the items in increasing order of s_i.
- Order the items in decreasing order of s_i.