Test 3

1. Consider the mixture sequence problem you encounter in Homework #1B and #1C. Do the following. (i) Jonny believes the mixture sequence problem is in NP. Is that true? Explain why or why not. (ii) Describe a polynomial-time reduction scheme to transform any give problem instance x for the mixture sequence problem into a corresponding problem instance y for the knapsack problem such that a solution to y can lead to a solution to x. (iii) Since the knapsack problem is NP Complete, Jonny believes that the polynomial-time reduction scheme above shows that the mixture sequence problem is NP-Complete too. Is that true? Explain why or why not.

2. Consider the mixture sequence problem you encounter in Homework #1B and #1C. again. Do the following. (i) Describe a dynamical-programming algorithm to solve the problem in O(m*n) time where n is the length of the sequences and m is the target sum of the mixture sequence. (ii) Analyze the running time of your algorithm and show that it is indeed a O(m*n) time algorithm. (iii) Implement your algorithm into a program that can allow user to enter an problem instance and your program should then solve the problem instance to provide an answer to whether a mixture sequence ending in the target sum does exists. Your program should print out a mixture sequence ending in the target sum if such a mixture sequence does exist.

3. Consider any three given strings composed of of English letters: $X = x_1x_2\ldots x_m$, $Y = y_1y_2\ldots y_n$, $Z = z_1z_2\ldots z_{m+n}$. The string Z is a shuffle of X and Y if and only if Z can be formed by interspersing the characters from X and Y in a way that preserves the left-to-right ordering of the characters from X and Y respectively. For example, **cchocohilaptes** is a shuffle of **chocolate** and **chips**, but **chcccochilatspe** is not. **(i)** Devise and describe a O(m*n) time dynamical programming algorithm that can always correctly determine whether Z is a shuffle of X and Y given any strings X,Y, and Z. **(ii)** Analyze the running time of your algorithm and show that it is indeed a O(m*n) time algorithm. **(iii)** Implement your algorithm into a program that can allow user to enter an 3 strings X, Y, Z and your program should then solve the problem instance to provide an answer to whether Z is a shuffle of X and Y. Your program should print out an explanation of the alignment of each character in Z with a unique character in X or Y if Z is indeed a shuffle of X and Y.

4. Your friend is working as a camp counselor, and he is in charge of organizing activities for a set of junior-high-school-age campers. One of his plans is the following mini-triathlon exercise: each contestant must swim 20 laps of a pool, then bike 10 miles, then run 3 miles. The plan is to send the contestants out in a staggered fashion, via the following rule: **the contestants must use the pool one person at a time**. In other words, first one contestant swims the 20 laps, gets out, and starts biking. As soon as this first person is out of the pool, a second contestant begins swimming the 20 laps; as soon as the second person is out and starts biking, a third contestant begins swimming, . . ., and so on. Note that multiple persons may be biking or running at the same time.

Each contestant has a projected swimming time (the expected time it will take him or her to complete the 20 laps), a projected biking time (the expected time it will take him or her to complete the 10 miles of bicycling), and a projected running time (the time it will take him or her to complete the 3 miles of running). Your friend wants to decide on a schedule for the triathlon: an order in which to sequence the starts of the contestants. Let's say that the completion time of a schedule is the earliest time at which all contestants will be finished with all three legs of the triathlon, assuming they each spend exactly their projected swimming, biking, and running times on the three parts. (Again, note that participants can bike and run simultaneously, but at most one person can be in the pool at any time.)

How would you determine the best order for sending n persons out if you want the whole competition to be over as early as possible? For example, see Problem instance #I of 3 persons and Problem instance #II of 10 persons below.

Problem instance #I	Swimming time	Biking time	Running time
Person #1	20	25	20
Person #2	35	20	15
Person #3	40	20	30

Problem instance #II	Swimming time	Biking time	Running time
Person #1	20	25	20
Person #2	35	20	15
Person #3	40	20	30
Person #4	25	30	25

Person #5	22	27	18
Person #6	37	18	17
Person #7	42	22	28
Person #8	27	32	24
Person #9	31	16	18
Person #10	36	24	26

(i) For Problem instance #I, manually enumerate all the possible orders of sending out these three persons and find out what is the best order to minimize the completion time.

(ii) Write a computer program that can read in the data of all the contestants and find the best order to minimize the completion time l by enumerating all the possible orders of sending out these persons. Use your program to solve both Problem instance #I and Problem instance #II. Report the optimal solutions (the schedules of the persons) for these two cases respectively. **Note that if you are able to implement the correct greedy algorithm described in (iii) and (iv) below, you will automatically get the points for (ii) here. In other words, you can skip (ii) if you can do (iii) and (ivv) correctly.**

(iii) Instead of the exhaustive search approach, there is actually a greedy criterion that can guide you to find the optimal solution for any give problem instance. Find such a greedy criterion and use the exchange argument to prove that it always leads to an optimal solution.

(iv) Implement the greedy algorithm based on the greedy criterion above into a program that can allow user to enter a problem instance and your program should then solve the problem instance to print out an optimal schedule.