1. Find a longest common subsequence of <1,0,0,1,0,1,0,1,0,0,1,1,0,1,1> and <0,1,0,1,1,0,1,1,0,1,1,1,0,1>.

I used the algorithm that we learned in class of constructing a table, and then following a path through the table to find the optimal solution and programmed this into the LCS.java program included with this submission. I used https://www.geeksforgeeks.org/printing-longest-common-subsequence/?ref=rp to understand how to make the program get the optimal chain using the table.

Using this code, we can obtain:

	Y	1	0	0	1	0	1	0	1	0	0	1	1	0	1	1
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2
0	0	1	2	2	2	3	3	3	3	3	3	3	3	3	3	3
1	0	1	2	2	3	3	4	4	4	4	4	4	4	4	4	4
1	0	1	2	2	3	3	4	4	5	5	5	5	5	5	5	5
0	0	1	2	3	3	4	4	5	5	6	6	6	6	6	6	6
1	0	1	2	3	4	4	5	5	6	6	6	7	7	7	7	7
1	0	1	2	3	4	4	5	5	6	6	6	7	8	8	8	8
0	0	1	2	3	4	5	5	6	6	7	7	7	8	9	9	9
1	0	1	2	3	4	5	6	6	7	7	7	8	8	9	10	10
1	0	1	2	3	4	5	6	6	7	7	7	8	9	9	10	11
1	0	1	2	3	4	5	6	6	7	7	7	8	9	9	10	11
0	0	1	2	3	4	5	6	7	7	8	8	8	9	10	10	11
1	0	1	2	3	4	5	6	7	8	8	8	9	9	10	11	11

LCS of {0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1} and {1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1} is {1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1} and the length of this sequence is 11.

The sequence that we get depends on the order in which the sequences are entered. If we have X = {0,1,0,1,1,0,1,1,0,1,1,1,0,1} and Y = {1,0,0,1,0,1,0,1,0,0,1,1,0,1,1}, then we get the above result of {1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1}. However, if we have Y = {0,1,0,1,1,0,1,1,0,1,1,1,0,1}, X = {1,0,0,1,0,1,0,1,0,0,1,1,0,1,1}, we get {0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1} as our common subsequence.

If we look at the input sequences, we find that both {1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1} and {0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1} are valid common subsequences of length 11.

2. Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is A_1(5x10), A_2(10x3), A_3 (3x12), A_4(12x5), A_5(5x50), A_6(50x6), A_7(6x2).

I used the pseudocode for the algorithm we learned in class with a slight modification that I found at https://www.geeksforgeeks.org/printing-matrix-chain-multiplication-a-space-optimized-

solution/?ref=rp in order to create the MatrixChainMulti.java program included with this submission. This change allows the program to easily print a list of how the matrices will be parenthesized using recursion.

Our first table stores the number of multiplications:

0	150	330	405	1655	2010	1452
	0	360	330	2430	1950	1352
		0	180	930	1770	1292
			0	3000	1860	1220
				0	1500	1100
					0	600
						0

Our second table stores how to split the matrices

0						
0	1					
0	2	2				
0	2	2	3			
0	4	2	4	4		
0	2	2	4	4	5	
0	1	2	3	4	5	6

In the program, these two tables are stored as 2 halves of the same table in order to save space.

By following these tables or running the code, we get:

$(A_1 \times (A_2 \times (A_3 \times (A_4 \times (A_5 \times (A_6 \times A_7))))))$ as our optimal parenthesization with total number of multiplications 1452. (The code writes the matrices using letters A-G instead of A_1-A_7 but the result is the same).

3. Extra Credit: Prove that Dijkstra's algorithm provides an optimal solution to the shortest path problem in a graph.

Proof that Dijkstra's Algorithm returns the optimal solution.

Let d(v) be the shortest distance from the source to vertex v found by Dijkstra's algorithm.

Let s(v) be the optimal shortest distance from the source to vertex v.

If Dijkstra's algorithm is optimal, then for all v_i in graph G, $d(v_i) = s(v_i)$.

We will prove this by induction using a graph G which has n vertices.

Our base case is when we use Dijkstra's algorithm on just one vertex of G. In this case, $d(v_1) = 0$ and $d(v_1) = 0$ since if there is only one vertex to look at, then v_1 = source and no travelling is done.

For our inductive hypothesis, assume for the first n-1 vertices in G, $d(v_i) = s(v_i)$ for $i = 1, \ldots, n-1$.

Using our inductive hypothesis, we need to show that for vertex v_n, $d(v_n) = s(v_n)$. We know by the inductive hypothesis that for $i = 1, \ldots, n-1$, $d(v_i)$ is the correct minimum distance from source to v_i and therefore, $d(v_n)$ can be found by choosing some edge uv_n such that if $d(v_n) = d(u) + \text{length}(uv_n)$, then $d(v_n)$ is minimized.

Suppose for contradiction that $s(v_n) < d(v_n)$. If this is true, then one of two cases must be true:

1. There exists a shorter path to get to u from source. Since $u \neq v_{n+1}$, then by our inductive hypothesis, $d(u) = s(u)$, so this shorter path cannot exist, and this case is impossible.
2. There is another edge, wv_n such that $d(w) + \text{length}(wv_n)$ is smaller than $d(u) + \text{length}(uv_n)$, but if such an edge existed, it must have already been checked by Dijkstra's algorithm and would have been chosen, so this case is also impossible.

Therefore, it must be true that $s(v_n) = d(v_n)$, so by induction, Dijkstra's algorithm must find the optimal solution for every vertex in G.