Practice Problem Set 10

. Problem 1

In class we saw the dynamic programming solution for Longest Palindrome Substring, which produces the table $L[i, j]$ where $L[i, j]$ is defined as 1 if the substring $s[i, j]$ is a palindrome, and 0 otherwise. The table dimensions are $n \times n$. Using the results of a DP table, describe an algorithm to output **all** palindrome substrings of maximum length, and justify its runtime of $\Theta(n^2)$. *Note: You must provide a general algorithm. The table below is just an example DP table for $L[i, j]$ where $n = 9$*

	1	2	3	4	5	6	7	8	9
1	1	0	1	0	0	0	0	0	0
2		1	0	0	0	0	0	0	0
3			1	0	0	1	0	0	0
4				1	1	0	0	0	0
5					1	0	0	0	0
6						1	0	1	0
7							1	0	1
8								1	0
9									1

Problem 2

In class we studied the Longest Palindrome Subsequence. Write a procedure (with pseudo-code) that takes as input the completed DP table and the initial strings, and outputs the longest palindrome Subsequence.

Problem 3

A set of n items is such that each item has a specific weight, w_i, for $1 \leq i \leq n$. We would like to find a subset of those items that has total weight T (if there is one). Describe a brute-force algorithm for this problem and determine its runtime. Describe a recursive solution (with pseudo-code) for this problem. Next, provide a dynamic programming solution to this problem and explain the runtime. Describe how to use the DP table to output the elements whose total weight is T.

Problem 4

Suppose each of the items in Problem 3 (above) has an associated *value*, v_i, and also a weight w_i. Provide a dynamic programming solution that finds a subset of the items with maximum value, such that the total weight is at most T and explain the runtime. Can you also output the specific elements which correspond to the maximum value?

Problem 5

Suppose you are going on a long bike ride with your e-bike. You must ride exactly n miles. You will need to stop to replace your battery along the way. The bike trail is lined with battery pick-up stations, where you can **trade your battery for a new charged battery**. There is one battery station at every mile marker. However, the batteries at each station vary. For example, suppose that at station 1 they have batteries that last 3 miles, whereas station 2 may have batteries that last 7 miles. The goal is to complete the bike ride using the **minimum number of stops to replace your battery**. You cannot run out of power! If you pick up a battery pack worth 4 miles, then you cannot bike more than 4 miles!

The input to the problem is an array b[0...n], where b[i] stores the battery life of the batteries at station i. The output is the minimum number of stops required. Assume you start the trail at mile 0, and that you receive the battery at station 0. Provide a dynamic programming solution to this problem, and determine the runtime of your algorithm.