COP 3503H – HW 4

Question 1:

1. A longest palindrome subsequence is the longest subsequence of a string that has the same letters when read forward and backward.
2. To solve this problem recursively, we will first compare the (i) first and (j) last characters to see if they match. If they do, do a recursive call on the string not including the first and last characters (from i+1 to j-1) and add 2 to the total length of the palindrome subsequence. If they do not match, do a recursive call on the string not including the first, and the string not including the last (from i+1 to j and i to j+1) and pick the one with the larger palindrome subsequence to continue working with. If a letter is being compared to itself, return one, and if two adjacent letters are the same, return two.  
   This can also be solved using dynamic programming. A two-dimensional array will be created to store the results of each subproblem. First, the positions in the two-dimensional array where a character is compared to itself are filled. Then, the string will be inspected in distances (dist) from size 2 to length, comparing characters dist apart. As in the recursive solution, when a matching pair is found, it will be added to the previous best. If a matching pair is not found, the previous best is used. Once all values of dist are inspected, the length of the longest palindromic subsequence is returned.
3. Longest\_Palendrome(string, length)  
    Let P[length, length]  
    for i = 0 to length  
    P[i,i] = 1  
    Let dist be the distance between characters being analyzed, starting at 2 and going to length  
    for i = 0 to length - dist  
    let j be the character dist from i  
    if (string[i] == string[j] AND dist == 2)  
    P[i,j] = 2  
    else if (string[i] == string[j])  
    P[i,j] = P[i+1, j-1] + 2  
    else P[i,j] = max(P[i,j-1], P[i+1,j])  
    return P[0, length-1]
4. Generate\_LPS(string, maxLen, P)  
    Let pal be a string of length length  
    Let i = 0, j = length - 1  
    while (P[i,j] != 0)  
    if (P[i,j] > P[i+1,j] AND P[i,j] > P[i,j-1])  
    pal = pal + string[j]  
    i++  
    j--  
    else if (P[i+1,j] > P[i,j-1])  
    i++  
    else j--  
    Let i = 0  
    while pal[i] != null character  
    print pal[i]  
    i++  
    if maxLen is odd  
    i--  
    while i > 0  
    print pal[i]  
    i--
5. The running time for the algorithm is order n^2