Ryan Young

CSE 3500

11/21/19

Homework 9

1.) Adapt the algorithm so that it gives the value of the optimal solution to the knapsack problem instead.

```
Function KnapSackSol(n, W):
        Allocate M[1. . . n, 1. . . W]
        Initialize M[0, w] = 0 for each w 1 to W
        For I = 1 to n
                For w = 1 to w
                        If w < wi
                                M[I, w] = KnapSackSol(I – 1, W)
                        Else
                                M[I, W] = max(M[i-1, W], Vi + M[I-1, W-Wi])
                        End if
                End for
        End for
        Return M[n, W]
End function
```

My solution to adapt the given pseudo code to solve the knapsack problem instead is to add values instead of adding weights when the weight is a compatible one. This algorithm will fill the M array in O(nW) time.

2.) The 2d array contains the value of the optimal solution to each one of the subproblems. We have seen that the optimal set, the items that end up in your bag, can be found in linear time. Propose and algorithm that takes the array as an input and returns the optimal set. The algorithm also has access to the set of items and their associated weights and values.

```
Function KnapSackSet(M, n):
        result = M[n, W]
        w = W
        S = {} //empty set
        for (i = n; i > 0 and result > 0; i-= 1)
                If (result == M[i - 1 , w]);
                else{
                        add (i-1) to S
                        result = result - V[i - 1];
                        w = w – W[i - 1];
                end if
        end for
        return S
```

End Function

To make this function run in linear time I loop through each object in the double array M that is passed in. Then using the values, I already have from the array M that is passed in I find the optimal set in a similar fashion to number 1.