W = max weight capacity
w = weight array
v = value array.

Answer 4 a)

```
int[] func knapsack (W, v[], w[])
    int max_Capacity = W;
    int current_weight = 0;
    int knapsack = [];
    int tempWeight = 0;
    for each i in v.length {
        if ((currentWeight + w[i]) < max_Capacity) {
            current_weight = current_weight + w[i];
            knapsack.add (w[i], v[i]);
        }
    }
    return knapsack;
}
```

b) Run-time efficiency of algorithm = O(n)

c) The problem will yield an optimal solution as due to the conditions promised in the question itself where we assume an array for both w and v to be sorted in a manner such that,

$w_1 \leq w_2 \leq w_3 \leq \dots w_n$

$v_1 \geq v_2 \geq \dots \geq_n$

The condition itself will yield optimal solution as we will be able to make most amount of profit (v) with carrying least weights as possible in the bag (w). This continues on adding items to bag as long as we stay under the maximum capacity of the bag. Thus, it proved that the algorithm will yield an optimal solution.