**Problem 1**

KnapSack (v, w, n, W)

for w = 0 to W

V[0,w] ← 0;

for i = 1 to n

for w = 0 to W

if(w[i] <= w)

V[i,w] ← max{V[i – 1, w], v[i] + V[i-1,w-w[i]]};

Keep[i, w] ← true;

else

V[i,w] ← V[i-1,w];

Keep[i, w] ← false;

for i = n down to 1

if(Keep[i, w] == true)

output: i >> W – w[i];

Retun V[n,W]

**Problem 2**

Time complexity: O(nW)

No. Programming approach of solving Knapsack problem has not changed the exponential time complexity of the original brute force solution.

**Problem 3**

In knapsack 0/1 problem, we need 2 inputs (1 array & 1 integer) to solve this problem:

* An array of n items: [n1, n2, n3...] - each item with its value index and weight index.
* An integer W is maximum acceptable weight

Let's assume n=10 and W=8:

* n = [n1, n2, n3... n10]
* W = 1000 in binary term (4-bit long)

So the time complexity T (n) = O (n\*W) = O (10\*8) = O (80)

1. If we double the size of n:

* n = [n1, n2, n3... n10] -> n = [n1, n2, n3 ... n20]

So time complexity T(n) = O (n\*W) = O (20\*8) = O (160)

1. If we double the size of W, it does not mean W=20, but the length is twice long:

* W = 1000 -> W = 10000000 in binary term (8-bit long)

So T (n) = O (n\*W) = O (10\*128) = O (10\*28) = O (1280)

So the runtime is more accurately said to be O (N.2bits in W), which is exponential