In [ ]:
```
Dynamic-0-1-knapsack (v, w, n, W)
for w = 0 to W do
    c[0, w] = 0
for i = 1 to n do
    c[i, 0] = 0
    for w = 1 to W do
        if wi ≤ w then
            if vi + c[i-1, w-wi] then
                c[i, w] = vi + c[i-1, w-wi]
            else c[i, w] = c[i-1, w]
        else
            c[i, w] = c[i-1, w]
```

In [4]:
```
def knapSack(W, wt, val, n):
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]

    # Build table K[][] in bottom up manner
    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif wt[i-1] <= w:
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]],  K[i-1][w])
            else:
                K[i][w] = K[i-1][w]

    return K[n][W]

val = [70, 90, 120]
wt = [10, 20, 30]
W = 50
n = len(val)
print(knapSack(W, wt, val, n))
```

```
210
```