#Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy

```
# code
def knapSack(W, wt, val, n):
    dp = [0 for i in range(W+1)] # Making the dp array
    for i in range(1, n+1): # taking first i elements
        for w in range(W, 0, -1): # starting from back,so that we also have
data of
            # previous computation when taking i-1 items
            if wt[i-1] <= w:
                # finding the maximum value
                dp[w] = max(dp[w], dp[w-wt[i-1]]+val[i-1])
    return dp[W] # returning the maximum value of knapsack

# Driver code
val = [60, 100, 120]
wt = [10, 20, 30]
W = 50
n = len(val)
print(knapSack(W, wt, val, n))
```

OUTPUT: