```
"""
Name: Anuj Mahendra Mutha
Class:BE-4 Computer Engineering
Batch : R4
Lab Assignment No: 04
Title: Write a program to solve a 0-1 Knapsack problem using dynamic
programming or branch and
bound strategy.
"""
# a dynamic approach
# Returns the maximum value that can be stored by the bag
def knapSack(W, wt, val, n):
   K = [[0 for x in range(W + 1)] for x in range(n + 1)]
   #Table 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]
#Main
val = []
wt = []
print("Enter Number of Items to be tested:")
num=int(input())
for i in range(num):
    print("Enter item value")
    v=int(input())
    print("Enter item weight")
    w=int(input())
    val.append(v)
    wt.append(w)
W = int(input("Enter Knapsack Capcity"))
n = len(val)
print("Maximum Profit: ",knapSack(W, wt, val, n))

'''
>> %Run BinaryKnapSack.py
Enter Number of Items to be tested:
3
Enter item value
60
Enter item weight
10
Enter item value
100
Enter item weight
20
Enter item value
120
Enter item weight
30
Enter Knapsack Capcity: 50
Maximum Profit: 220
'''
```