'''

Problem:

Given a value N, if we want to make change for N cents,

and we have infinite supply of each of S = { S1, S2, .. , Sm} valued coins,

how many ways can we make the change? The order of coins doesn’t matter

Ways:

way1: dynamic program

two properties of DP problem: (1) overlapping sub-problem;(2)optima sub-structure

we can divide solutions into two sets:

not contain mth coin, contain at least one mth conin

table[icha][jcoin] = table[icha][jcoin-1] + table[icha-coin\_j][jcoin]

construct the table from top to bottom, left to right

way2: optimize the way1 DP solution

way3: recursion solution

'''

def make\_change(s, m, n):

# create table

table = [[[0] for x in range(m)] for x in range(n + 1)]

# fill data for 0 change

for i in range(m):

table[0][i] = 1

# dynamic programming

for icha in range(1, n+1):

for jcoin in range(m):

# including object jcoin

x = table[icha - s[jcoin]][jcoin] if icha >= s[jcoin] else 0

# exclude object jcoin

y = table[icha][jcoin-1] if jcoin >=1 else 0

table[icha][jcoin] = x + y

return table[n][m-1]

s = [2,3,5,6]

m = len(s)

n = 10

ways = make\_change(s, m, n)

print(ways)