Dice throw

Given n dice each with m faces, numbered from 1 to m, find the number of ways to get sum X. X is the summation of values on each face when all the dice are thrown.

In the simple case of two throws of a six-sided die, the combinations of eyes that will give total of seven eyes are:

$$(6,1),(5,2),(4,3),(3,4),(2,5),(1,6)$$

So six ways.

To make it more general we can say that our dice are m-sided, that we have n dice and that we want a total of x eyes, and that the function that computes the number of ways to get x is called $W(m,n,x)$.

If the first die shows $e \in \{1,\ldots,m\}$ eyes, then the number of ways to get x eyes in total is $W(m,n-1,x-m)$. I.e., the number of ways the remaining $n-1$ dice produces $x-m$ eyes. Since the first die can show m different values, we need to sum over those to get the total number of ways. This produces the following recursive formula:

$$W(m,n,x) = \sum_{e=1}^{m} W(m,n-1,x-e)$$

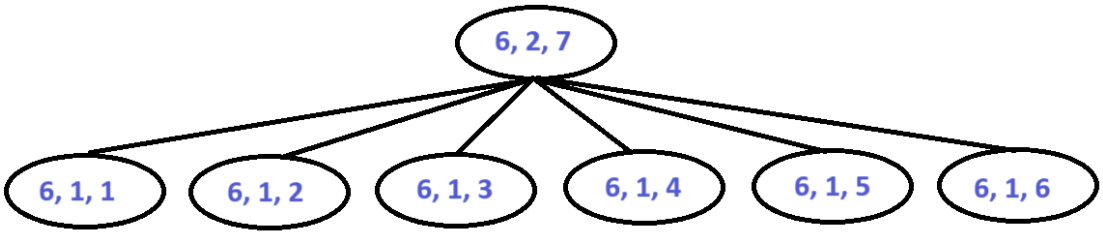
We need a base case for the recursion. In the formula above, each recursive call reduce the problem in not one but two ways (the number of dice and the total number of eyes). n is reduced by 1 and x is reduced by e. So we actually have two base cases:

- If the total we want is not at least 1. I.e., $x < 1$, then $W(m,n,x)$ is 0.
- If there is only one die and if that die shows m or fewer eyes then there is a single way to get x: So if $n = 1$ and $x \geq m$ then $W(m,n,x)$ is 1.

$$W(m,n-1,x-m) = \begin{cases} 0, & \text{if } x < 1 \\ 1, & \text{if } x \geq 1 \text{ and } n = 1 \text{ and } x \leq \\ \sum_{e=1}^{m} W(m,n-1,x-e), & \text{otherwise} \end{cases}$$

Exercise

Make sure you understand the redursion. Take a piece of paper and draw out the recursive function calls made when computing $W(6,2,7)$



Exercise

Implement the recursive solution in Python.

If you call your function like this `find_nr_ways(6, 2, 7)`, it should return `6`.

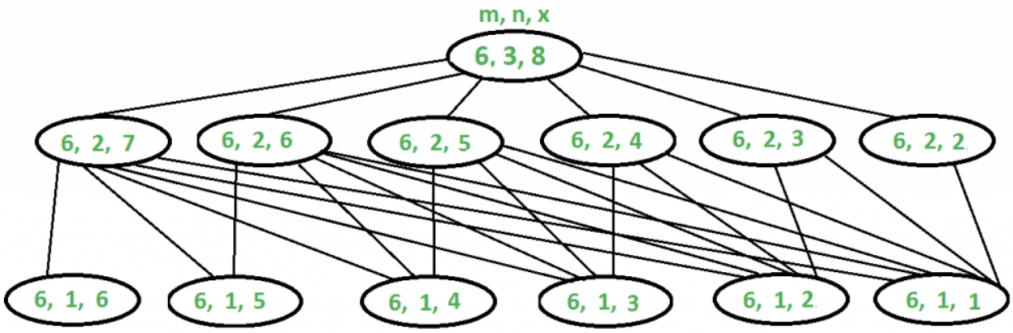
In [ ]:
```
def find_nr_ways(m, n, x):
    nr_ways = 0
    if x < 1: #basecase
        return 0
    elif x >= 1 and n == 1 and x <= m: #basecase, hvis x(sum) er større e
        return 1
    for e in range(1, m+1): #otherwise, ranger over alle sider(m) fra 1-m
        nr_ways += find_nr_ways(m,n-1,x-e)
    return nr_ways
    # Your code goes here...

find_nr_ways(6, 2, 7)
```

Out[ ]: 6

Exercise

The above problem exhibits overlapping subproblems. If there are three dice, each with 6 sides and we need to find the number of ways to get sum 8:



That may not seem like much of a problem, but try calling your function like this `find_nr_ways(20, 10, 100)` (yes a 20-sided die is a thing). It takes a long time. You can interrupt it by clicking the black square in the tool bar.

Still it was not for nothing, you have found a solution to your problem using recursion, now you can try to make it efficient using dynamic programming.

In [ ]:
```python
nr_ways_dict = {}

def find_nr_ways(m, n, x):
    nr_ways = 0
    if x<1:
        return 0
    elif x >=  1 and n == 1 and x <= m:
        return 1
    for e in range(1, m+1):
        key = '{},{},{}'.format(m, n-1, x-e)
        if key not in nr_ways_dict:
            nr_ways_dict[key] = find_nr_ways(m, n-1, x-e)
            #print(nr_ways_dict)
            nr_ways += nr_ways_dict[key]
        else:
            nr_ways += nr_ways_dict[key]
    print(nr_ways_dict)
    return nr_ways
    # Your code goes here...

find_nr_ways(10, 6, 8)
```

{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1

0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '10,4,3': 0, '10,3,-8': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '10,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '10,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0': 0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '10,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0': 0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4': 3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '10,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0, '10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 10, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '10,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0': 0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0, '10,5,6': 5, '10,4,-5': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4':

```
3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '1
0,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-
4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0,
'10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,
0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1
0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '1
0,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0':
0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0,
'10,5,6': 5, '10,4,-5': 0, '10,5,5': 1, '10,4,-6': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-
2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4':
3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '1
0,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-
4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0,
'10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,
0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1
0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '1
0,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0':
0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0,
'10,5,6': 5, '10,4,-5': 0, '10,5,5': 1, '10,4,-6': 0, '10,5,4': 0, '10,4,-
7': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-
2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4':
3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '1
0,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-
4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0,
'10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,
0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1
0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '1
0,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0':
0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0,
'10,5,6': 5, '10,4,-5': 0, '10,5,5': 1, '10,4,-6': 0, '10,5,4': 0, '10,4,-
7': 0, '10,5,3': 0, '10,4,-8': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-
2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4':
3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '1
0,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-
4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0,
'10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,
0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1
0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '1
0,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0':
0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0,
'10,5,6': 5, '10,4,-5': 0, '10,5,5': 1, '10,4,-6': 0, '10,5,4': 0, '10,4,-
7': 0, '10,5,3': 0, '10,4,-8': 0, '10,5,2': 0, '10,4,-9': 0}
{'10,1,3': 1, '10,1,2': 1, '10,1,1': 1, '10,1,0': 0, '10,1,-1': 0, '10,1,-
2': 0, '10,1,-3': 0, '10,1,-4': 0, '10,1,-5': 0, '10,1,-6': 0, '10,2,4':
3, '10,1,-7': 0, '10,2,3': 2, '10,1,-8': 0, '10,2,2': 1, '10,1,-9': 0, '1
0,2,1': 0, '10,2,0': 0, '10,2,-1': 0, '10,2,-2': 0, '10,2,-3': 0, '10,2,-
4': 0, '10,2,-5': 0, '10,3,5': 6, '10,2,-6': 0, '10,3,4': 3, '10,2,-7': 0,
'10,3,3': 1, '10,2,-8': 0, '10,3,2': 0, '10,2,-9': 0, '10,3,1': 0, '10,3,
0': 0, '10,3,-1': 0, '10,3,-2': 0, '10,3,-3': 0, '10,3,-4': 0, '10,4,6': 1
0, '10,3,-5': 0, '10,4,5': 4, '10,3,-6': 0, '10,4,4': 1, '10,3,-7': 0, '1
0,4,3': 0, '10,3,-8': 0, '10,4,2': 0, '10,3,-9': 0, '10,4,1': 0, '10,4,0':
0, '10,4,-1': 0, '10,4,-2': 0, '10,4,-3': 0, '10,5,7': 15, '10,4,-4': 0,
'10,5,6': 5, '10,4,-5': 0, '10,5,5': 1, '10,4,-6': 0, '10,5,4': 0, '10,4,-
7': 0, '10,5,3': 0, '10,4,-8': 0, '10,5,2': 0, '10,4,-9': 0, '10,5,1': 0,
'10,5,0': 0, '10,5,-1': 0, '10,5,-2': 0}
```

Out[ ]: 21

Exercise

All your function calls share the first paramter `m`, but vary in their values of `n` and `x`. So what you need is a table where you can fill in the number of ways ($W(m, n, x)$) for each combination of `n` and `x`.

However, to fill it in, you also need to figure out how to compute the value in each cell from values in other cells. Draw a table on paper, look at the figure above, and decide which set of other cells you need to compute each cell.

Once you figure that out, you should be able to in what order you will need to fill in the cells in the table.

Exercise

Implement a solution to the problem using dynamic programming

In [ ]:
```
import numpy as np

def print_dp_matrix(seq1, seq2, matrix):
    max_len = max(len(str(cell)) for row in matrix for cell in row)
    fmt = "{{:>{}}}".format(max_len+1)
    row_fmt = fmt * (len(matrix[0])+1) + '\n'
    mat_fmt = row_fmt *  (len(matrix)+1)
    seq1 = ' ' + seq1
    seq2 = ' ' + seq2
    lst = [' '] + list(seq2)
    for i in range(len(seq1)):
        lst.extend([seq1[i]] + list(map(repr, matrix[i])))
    print(mat_fmt.format(*lst))

def find_nr_ways(m,n,x):

    # Create a table to store results of subproblems. One extra
    # row and column are used for simplicity (number of dice
    # is directly used as row index and sum is directly used
    # as column index). The entries in 0th row and 0th column
    # are never used.

    # make an x+1 by n+1 table with all zeros
    table=[[0]*(n) for i in range(x)]
    #print(table)
    for row in range(x):
        for column in range(n):
            #print(row, column)
            if row == column:
                table[row][column] = 1
            elif column == 0 and row+1 < m:
                table[row][column] = 1
            else:
                table[row][column] = table[row-1][column] + table[row-1][

    print_dp_matrix("|"*(x-1), "-"*(n-1), table)
```

```
    return table[-1][-1]

find_nr_ways(6, 10, 15)
```

```
              _    _    _    _    _    _    _    _    _
        1     0    0    0    0    0    0    0    0    0
    |   1     1    0    0    0    0    0    0    0    0
    |   1     2    1    0    0    0    0    0    0    0
    |   1     3    3    1    0    0    0    0    0    0
    |   1     4    6    4    1    0    0    0    0    0
    |   1     5   10   10    5    1    0    0    0    0
    |   1     6   15   20   15    6    1    0    0    0
    |   1     7   21   35   35   21    7    1    0    0
    |   1     8   28   56   70   56   28    8    1    0
    |   1     9   36   84  126  126   84   36    9    1
    |   2    10   45  120  210  252  210  120   45   10
    |  12    12   55  165  330  462  462  330  165   55
    |  67    24   67  220  495  792  924  792  495  220
    | 287    91   91  287  715 1287 1716 1716 1287  715
    |1002   378  182  378 1002 2002 3003 3432 3003 2002
```

Out[ ]: 2002

In [ ]:
```
import numpy as np

def find_nr_ways(m,n,x):

    # Create a table to store results of subproblems. One extra
    # row and column are used for simplicity (number of dice
    # is directly used as row index and sum is directly used
    # as column index). The entries in 0th row and 0th column
    # are never used.

    # make an x+1 by n+1 table with all zeros
    table=[[0]*(x+1) for i in range(n+1)]

    #for alle hvor der kun er en terning
    for j in range(1, min(m+1,x+1)):
        table[1][j]=1

    for i in range(2, n+1):
        for j in range(1,x+1):
            for k in range(1,min(m+1, j)):
                table[i][j]+=table[i-1][j-k]
            #print(np.array(table))

    print(np.array(table))

    return table[-1][-1]

find_nr_ways(6, 10, 10)
```

```
[[  0   0   0   0   0   0   0   0   0   0   0]
 [  0   1   1   1   1   1   1   0   0   0   0]
 [  0   0   1   2   3   4   5   6   5   4   3]
 [  0   0   0   1   3   6  10  15  21  25  27]
 [  0   0   0   0   1   4  10  20  35  56  80]
 [  0   0   0   0   0   1   5  15  35  70 126]
 [  0   0   0   0   0   0   1   6  21  56 126]
 [  0   0   0   0   0   0   0   1   7  28  84]
 [  0   0   0   0   0   0   0   0   1   8  36]
 [  0   0   0   0   0   0   0   0   0   1   9]
 [  0   0   0   0   0   0   0   0   0   0   1]]
```

Out[ ]: 1