**PART B**

Now comes the real challenge. You were happily spending a lazy afternoon playing your board game with your dice when suddenly the mischievous Norse God Loki ( You love Thor too much & Loki didn’t like that much ) appeared.

Loki dooms your dice for his fun removing all the “Spots” off the dice. No problem! You have the tools to re-attach the “Spots” back on the Dice.

However, Loki has doomed your dice with the following conditions:

● Die A cannot have more than 4 Spots on a face.

● Die A may have multiple faces with the same number of spots.

● Die B can have as many spots on a face as necessary i.e. even more than 6. But in order to play your game, the probability of obtaining the Sums must remain the same! So if you could only roll P(Sum = 2) = 1/X, the new dice must have the spots reattached such that those probabilities are not changed.

Input:

● Die\_A = [1, 2, 3, 4, 5, 6] & Die B = Die\_A = [1, 2, 3, 4, 5, 6]

Output:

● A Transform Function undoom\_dice that takes (Die\_A, Die\_B) as input & outputs New\_Die\_A = [?, ?, ?, ?, ?, ?],New\_Die\_B = [?, ?, ?, ?, ?, ?] where,

● No New\_Die A[x] > 4

**Explanation:(Question)**

The problem statement portrays a situation where Loki has eliminated the spots from your dice, and you want to reattach the spots to both die A and die B. Loki gives some conditions. That includes:

* Die A cannot have more than 4 spots on a face.
* Die A may have multiple faces with the same number of spots.
* Die B can have as many spots on a face as necessary, even more than 6.

The challenge is to reattach the spots so that the likelihood of acquiring the aggregates while throwing the dice continues as before.

**Explanation:(Code)**

scaling\_factor: It is calculated as the sum of spots on die A divided by the sum of spots on die B. This scaling factor is used to adjust the values of die B based on the values of die A.

x: It initializes an empty list and then iterates through each element in die A, appending the minimum of that element and 4 to the list x. This ensures that die A has no more than 4 spots on the face.

y: It initializes another empty list and then iterates through each element in die B. For each element, it calculates a scaled value by multiplying it with the scaling factor, rounding to the nearest integer, and taking the minimum of that value and 6. This ensures that die B can have as many spots as necessary but no more than 6.

The function then returns the modified dice lists x and y. Then I defined two lists die\_a and die\_b representing the faces of two dice. Call the undoom\_dice function with the die\_a and die\_b lists and unpack the returned values into new\_die\_a and new\_die\_b. A. Finally, print the changed dice records new\_die\_a and new\_die\_b.

**Code:**

def undoom\_dice(die\_a, die\_b):

scaling\_factor = sum(die\_a) / sum(die\_b)

x=[]

for spots in die\_a:

x.append(min(4,spots))

y=[]

for spots in die\_b:

y.append(min(6,round(spots\*scaling\_factor)))

return x, y

die\_a = [1, 2, 3, 4, 5, 6]

die\_b = die\_a

new\_die\_a, new\_die\_b = undoom\_dice(die\_a, die\_b)

print("\nNew Die A:", new\_die\_a)

print("New Die B:", new\_die\_b)

**Output:**

