COSC326
Samuel Ng - 2955262
Sayed Latif - 8098066

The assignment states that:
We are given a supply of cubes of size 1x1x1.
These cubes are entirely blue or yellow.
We can have as many of each type as we want.

The goal is to form a 2x2x2 cube of 8 1x1x1 cubes.
These 2x2x2 cubes will have a pattern depending on the colours of the 1x1x1 cubes.
If any of these patterns can be achieved through rotation, they might as well be the same pattern.

In the below example, we can assume that 1 is yellow and 0 is blue.

This implies that if we have a 2x2x2 cube of:
1 1
0 0

0 0
0 0
It might as well count for any possible rotation that can be achieved. For example:
0 0
1 1

0 0
0 0

To achieve this, we need to simulate the 2x2x2 cube, and all possible variations available.
For example:
A cube of 0 blue and 8 yellow cubes.
A cube of 1 blue and 7 yellow cubes.
… A cube of 8 blue and 0 yellow cubes.

These variations do not yet generate any patterns, as they only simulate the number of 1x1x1 cubes in any 2x2x2 cube.

To generate patterns, we utilise permutations. Through permutations, we will be able to generate all unique patterns in the array, when viewed linearly. For example:
In a cube of 1 blue and 7 yellow cubes:
(For simplicity's sake, the array shall be represented in 8 values in an array)
1, 0, 0, 0, 0, 0, 0, 0 is one permutation
0, 1, 0, 0, 0, 0, 0, 0 is another
And this repeats until we are able to generate all possible permutations.

According to the assignment, if a pattern matches another through rotation, they may as well be the same. Thus, we first need to know how many possible rotations there are, given a cube's faces.

A cube has 6 faces, and each of those faces has 4 possible orientations (orientation here refers to the direction a face is rotated by, I.E., 0, 90, 180, 270). Multiplied by the number of faces a cube has, this gives up 24 unique rotational values for any pattern.

With the above, we are able to:
Generate any pattern through permutations of any variation of 1x1x1 cubes.
IE:
For a variation of
1, 1, 0, 0, 0, 0, 0, 0
A possible pattern(through permutations) is:
1, 0, 0, 0, 1, 0, 0, 0
Which can be found through generated rotations, implying that both should count for the same pattern.

To confirm this, we used python. Python has numpy, a class that contains multiple methods to generate arrays and simulate rotations. We made use of the array method in numpy to generate a three dimensional integer array, initalized to 0, which let us represent our 2x2x2 cubes in code. To allow for the creation of 2x2x2 cubes with different 1x1x1 cubes, we created a method that changed values in the three dimensional array from 0 to 1. Through this method, we successfully created a simulation that could represent the first part: A 2x2x2 cube that was made up of any number of yellow or blue 1x1x1 cubes.

The next part to our simulation was to generate all unique permutations as listed above:
For a cube of 1 blue and 7 yellow:
1, 0, 0, 0, 0, 0, 0, 0 counts as 1 unique permutation
0, 1, 0, 0, 0, 0, 0, 0 counts as another, and so on.

To generate all possible unique permutations, we used another tool from python, called itertools. Itertools is another class that can generate all permutations without repetition, like how the example of 1 blue and 7 yellow illustrates, through the use of its permutations method. Through this, we successfully generated all the possible combinations/patterns that could be created from a 2x2x2 cube made of any number of yellow or blue 1x1x1 cubes. To verify that this indeed generated all unique combinations, we used the 8th line of pascal's triangle, listed here as :
1,8,28,56,70,56,28,8,1

The reason for using pascal's triangle is due to its relation to permutations; the entirety of the triangle tells us the number of ways to choose items given different parameters. In our case, we simply needed the 8th line, as we only had 2 different colours and 8 total 1x1x1 cubes, as shown in the example arrays. The line in pascal's triangle matched up with the number of unique permutations generated as code, confirming that it was indeed generating unique permutations. We did this as we did not know if itertools truly generated all unique permutations.

The next problem to tackle was the rotation values. Numpy contains a method called rot90, which allows us to rotate an array. Furthermore, this method contains an axis parameter, which allows us to rotate multiple(1d, 2d, 3d, etc) dimensional arrays in

different ways. This became the base point to generate 24 unique rotations. We found code online that generated all 24 rotations, and verified this with numpy's rot90 documentation. The author has code that verifies each rotation is unique to provide proof as well. An example of a rotation can be shown above, but shall be illustrated here for easier understanding using a two dimensional array:
1 1
0 0

after rotation:

1 0
1 0

After rotation:

0 0
1 1

Etc.

From this point we have:
A way to generate any 2x2x2 cube from any number of yellow or blue 1x1x1 cubes.
A way to generate all unique permutations without repetition from those 2x2x2 cubes.
A way to generate all rotational values of any unique permutation.

Our next step is to compare and contrast all our values. As the assignment states:
We have to ensure that we have found all the values, and not counted any arrangement twice.
An arrangement can be a pattern generated through rotational values, and thus should only count once.

Thus, we:
Generate all possible variations of 2x2x2 cubes.
IE:
0 blue, 8 yellow cubes
1 blue, 7 yellow cubes
Generate all possible permutations of these 2x2x2 cubes, based upon the number of blue or yellow cubes.
IE:
For 1 blue, 7 yellow cubes in a 2x2x2 cube,
1, 0, 0, 0, 0, 0, 0, 0
0, 1, 0, 0, 0, 0, 0, 0
0, 0, 1, 0, 0, 0, 0, 0

Generate all rotational values for EACH permutation.
IE: Looking from top down(or from the first four values in the above array)
Original:
1 0
0 0

Rotate90:

0 1
0 0

Rotate180:

0 0
0 1

Rotate270:

0 0
1 0

Compare these rotational values back to their respective specific permutation. For example:
In a cube represented by:
1, 0, 0, 0, 0, 0, 0, 0
Rotating it would effectively make it:
0, 1, 0, 0, 0, 0, 0, 0
Which exists in our generated lists of permutations. Thus, we do not count this value. Repeated for every single permutation and rotation, we would be able to get all possible variants of 2x2x2 cubes, using any combination of blue and yellow cubes, without counting any arrangement twice.

Our final answer for the problem is thus, 23.

References:
https://stackoverflow.com/questions/33190042/how-to-calculate-all-24-rotations-of-3d-array
https://docs.python.org/3/library/itertools.html
https://numpy.org/doc/stable/reference/generated/numpy.rot90.html
https://numpy.org/doc/stable/reference/generated/numpy.array.html
https://en.wikipedia.org/wiki/Pascal%27s_triangle