**Cube Fun**

**Ground Rules**

1. Must make a commit for your work to be considered (timestamp of commit will be used in “tie” situations)
2. Google all you want on this challenge, its original from the depths of Travis’ brain.
3. You can use any language that you see fit.
4. The implementation must be able to run with provided input. (run locally or deployed somewhere).
5. Postman will be used to validate, so you must have this on your machine.

**Assumptions:**

1. 0 will not be used as a value outside of the usage to identify which block to find adjacent blocks.
2. Values in blocks will be 1-9 except for the one 0 for the identification block.
3. Max block size will be 3 slices, 3 rows, 3 columns.

**Problem to Solve**

Create a webservice that takes a JSON representation of a DataCube (Slice, Rows, Columns) and determine calculates the SUM of the adjacent blocks (blocks that share a “side”) to the “0” value.

**AC1: One Row (CubeFunAC1.txt for request).**

Input: 1, 0, 3

Output: 1 + 3 = 4 (1 and 3 are the adjacent blocks)

**AC2: Two Row (CubeFunAC2.txt for request)**

Input: 1, 2, 3

4, 0, 6

2 + 4 + 6 = 12

2 is the top neighbor, 4 is the left neighbor, 6 is the right neighbor.

**AC3: 3 Row (CubeFunAC3.txt for request)**

Input: 1, 2, 3

4, 0, 6

7, 8, 9

2 + 4 + 6 + 8 = 20

**AC4: 3Dimensional (CubeFunAC4.txt for request)**

Slice 0:

1, 2, 3

4, 0, 6

7, 8, 9

Slice 1 (behind slice 0):

1, 2, 3

4, 5, 6

7, 8, 9

2 + 4 + 6 + 8 + 5 (from depth 1) = 25

**AC5: Full Cube (3 row, 3 column, 3 slices) (CubeFunAC5.txt for request)**

Input:

Slice 0:

1, 2, 3

4, 5, 6

7, 8, 9

Slice 1:

9, 8, 7

6, 0, 4

3, 2, 1

Slice 2:

1, 2, 3

4, 5, 6

7, 8, 9

6 + 5 + 5 + 4 +8 + 2 = 30

6 is left neighbor

5 is back neighbor

5 is front neighbor

4 is right neighbor

8 is top neighbor

2 is bottom neighbor

AC6:………….Another full cube that we’ll use as validation.