Student: Anushka Polapally
Username: K1330129
ID#: 41365

USA Mathematical Talent Search

Year	Round	Problem
36	2	1

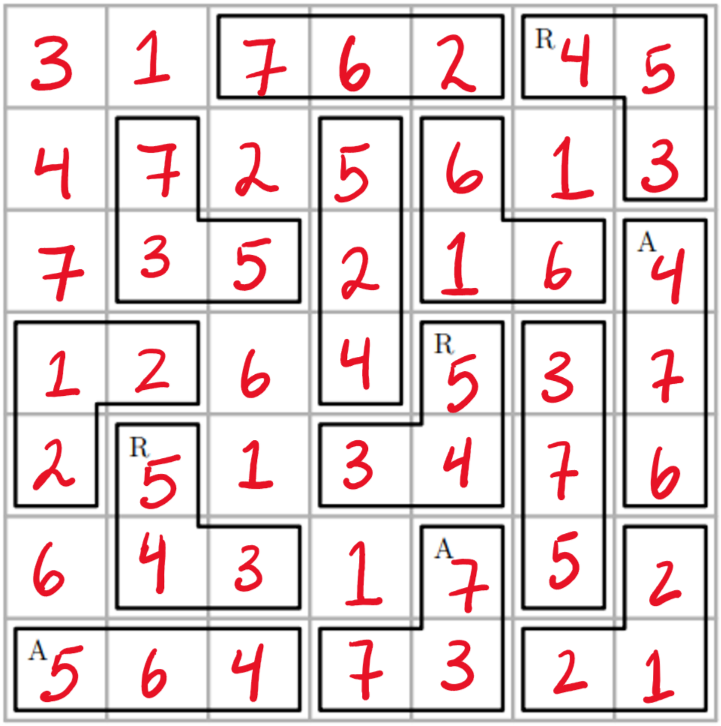

Figure 1: Enter Caption

Solution 1

Using the property that rows and columns are equal, we can create systems of equations to find the variables on which the rest of the grid is dependent. Using this method, we find that the grid is determined by 3 variables.

This C++ code below is used to search through all possible combinations of these 3 variables that determine the rest of the grid and finds the unique grids that satisfy the given conditions.

From the output given by the below code, we see that there are $\boxed{3}$ valid grids.

```cpp
#include <iostream>
#include <vector>
#include <array>
#include <set>
#include <cmath>

using namespace std;

// Function declarations
array<array<int, 3>, 3> rotate(const array<array<int, 3>, 3>& matrix);
array<array<int, 3>, 3> reflect(const array<array<int, 3>, 3>& matrix);
bool isInCombinations(const vector<array<array<int, 3>, 3>>& combinations, const array<array<int, 3>, 3>& matrix);

int main() {
    vector<array<array<int, 3>, 3>> combinations;

    for (int d = 1; d <= 12; d++) {
        for (int f = 1; f <= 12; f++) {
            for (int h = 1; h <= 12; h++) {
                array<array<double, 3>, 3> grid = {{
                    {(f + h) / 2.0, static_cast<double>(d + f - h), (d + h) / 2.0},
                    {static_cast<double>(d), (d + f) / 2.0, static_cast<double>(f)},
                    {(d + 2 * f - h) / 2.0, static_cast<double>(h), (2 * d + f - h) / 2.0}
                }};

                bool isInteger = true;
                bool containsOne = false;
                bool containsTwo = false;
                bool withinRange = true;
                set<double> uniqueNumbers;

                for (int i = 0; i < 9; i++) {
                    double current = grid[i / 3][i % 3];
                    uniqueNumbers.insert(current);

                    if (current == 1) containsOne = true;
                    if (current == 2) containsTwo = true;
                    if (floor(current) != current) isInteger = false;
                    if (current < 1 || current > 12) withinRange = false;
                }

```

Figure 2: Enter Caption

```
                if (isInteger && containsOne && containsTwo && withinRange && uniqueNumbers.size() != 9) {
                    array<array<int, 3>, 3> intGrid;
                    for (int i = 0; i < 9; i++) {
                        intGrid[i / 3][i % 3] = static_cast<int>(grid[i / 3][i % 3]);
                    }

                    if (!isInCombinations(combinations, intGrid)) {
                        cout << "[";
                        for (const auto& row : intGrid) {
                            cout << "[";
                            for (int val : row) {

                            }  string-literal
                            cout  Type: const char[2]
                        }     Size: 2 bytes
                        cout << "]" << endl;

                        combinations.push_back(intGrid);
                        combinations.push_back(rotate(intGrid));
                        combinations.push_back(rotate(rotate(intGrid)));
                        combinations.push_back(rotate(rotate(rotate(intGrid))));
                        combinations.push_back(reflect(intGrid));
                        combinations.push_back(reflect(rotate(intGrid)));
                        combinations.push_back(reflect(rotate(rotate(intGrid))));
                        combinations.push_back(reflect(rotate(rotate(rotate(intGrid)))));
                    }
                }
            }
        }
    }

    return 0;
}

array<array<int, 3>, 3> rotate(const array<array<int, 3>, 3>& matrix) {
    array<array<int, 3>, 3> rotated{};
    for (int i = 0; i < 9; i++) {
        rotated[i % 3][2 - i / 3] = matrix[i / 3][i % 3];
    }
    return rotated;
```

Figure 3: Enter Caption

```
}

array<array<int, 3>, 3> reflect(const array<array<int, 3>, 3>& matrix) {
    array<array<int, 3>, 3> reflected{};
    for (int i = 0; i < 9; i++) {
        reflected[i / 3][2 - i % 3] = matrix[i / 3][i % 3];
    }
    return reflected;
}

bool isInCombinations(const vector<array<array<int, 3>, 3>>& combinations, const array<array<int, 3>, 3>& matrix) {
    for (const auto& comb : combinations) {
        if (comb == matrix) {
            return true;
        }
    }
    return false;
}

```

Figure 4: Enter Caption

stdout stderr compile output scribble

```
[[2 3 1 ] [1 2 3 ] [3 1 2 ] ]
[[4 3 2 ] [1 3 5 ] [4 3 2 ] ]
[[5 5 2 ] [1 4 7 ] [6 3 3 ] ]
```

Successful, 0.00s, 3416KB

Student: Anushka Polapally
Username: K1330129
ID#: 41365

USA Mathematical Talent Search

Year	Round	Problem
36	2	4

Let the indexed alternating sum of T_i be A_i

The expected value of A can be defined as $\sum A_i P_i$, where P_i is the probability A_i occurs.

Since each unique A corresponds to a randomly chosen unique T, and each T has equal probability of being chosen, all A's have the same probability of being chosen.

There are 2^n ways to choose the subset T, and thus 2^n values of A. Thus $P_i = \frac{1}{2^n}$ for all i

This makes the expected value equivalent to:

$$\frac{\sum_{i=1}^{2^n} A_i}{2^n}$$

We can find $\sum_{i=1}^{2^n} A_i$ by grouping terms with a common x_k. Let each group of terms with a specific x_k be Z_k.

$$\sum_{i=1}^{2^n} A_i = \sum_{k=1}^{n} Z_k$$

We can now construct Z_k:

Since x_k can only be in positions $(1, \dots, k)$ we can calculate the number of x_k that will be in each of these positions.

If x_k is in position w, there must be $w-1$ x_j's, $j<k$ present in the subset. x_k must also be present! There are $\binom{k-1}{w-1}$ combinations of x_j's, $j<k$ and 2^{n-k} subsets with this combination and x_k present.

Thus $$Z_k = x_k \sum_{w=1}^{k} (-1)^{w+1} \binom{k-1}{w-1} 2^{n-k} w$$

$$Z_k = x_k 2^{n-k} \sum_{w=1}^{k} (-1)^{w+1} \binom{k-1}{w-1} w$$

Lemma 1: All $Z_i = 0$, $i \geq 3$

Proof of Lemma 1:

$$\sum_{i=0}^{n} (-1)^n \binom{n}{i} = \binom{n}{0} + \sum_{i=1}^{n-1} (-1)^i \binom{n}{i} + (-1)^n \binom{n}{n}$$

$$= \binom{n}{0} + \sum_{i=1}^{n-1} (-1)^i \left(\binom{n-1}{i-1} + \binom{n-1}{i}\right) + (-1)^n \binom{n}{n}$$

$$= \binom{n}{0} - \sum_{i=1}^{n-1}\left((-1)^{i-1}\binom{n-1}{i-1} - (-1)^{i}\binom{n-1}{i}\right) + (-1)^n\binom{n}{n}$$

$$= \binom{n}{0} - (-1)^{1-1}\binom{n-1}{1-1} + (-1)^{n-1}\binom{n-1}{n-1} + (-1)^n\binom{n}{n}$$

$$= 1 - 1 + (-1)^{n-1} - (-1)^{n-1}$$

Since

$$z_k = x_k 2^{n-k} \sum_{w=1}^{k} w \sum_{w=1}^{k} (-1)^{w+1}\binom{k-1}{w-1}$$

$z_3, z_4, \ldots = 0$ by Lemma 1

Since only z_1 and z_2 are nonzero, we calculate that

$$z_1 = x_1 \cdot 2^{n-1}$$

$$z_2 = -x_2 \cdot 2^{n-2}$$

So the expected value will be

$$\frac{x_1 \cdot 2^{n-1} - x_2 \cdot 2^{n-2}}{2^n} = \frac{2x_1 - x_2}{2^2} = \boxed{\frac{2x_1 - x_2}{4}}$$

Let $5^{1/3} = x$ so the given equation can be rewritten as $P(x + x^2) = 2x + 3x^2$

In order for the term $2x$ to be preserved in the polynomial, there must be a linear term with a coefficient of 2. Our polynomial P can now be written as $P(x) = Q(x) + 2x$ so, $P(x + x^2) = Q(x + x^2) + 2x + 2x^2 = 2x + 3x^2$

$$Q(x + x^2) = x^2$$

$Q(x)$ must also have a linear term for x^2 to be preserved so $Q(x+x^2) = K(x+x^2)+x+x^2 = x^2$ so

$$K(x + x^2) = -x$$

If we repeat this process again we have $G(x + x^2) - x^2 - x = -x$ so

$$G(x + x^2) = x^2$$

We see that the functions G and Q both intersect at the same point which restarts our cycle of polynomial decomposition showing that this cycle of polynomial decomposition goes on indefinitely and thus $P(x)$ cannot be decomposed in a finite number of polynomials, making it nonexistent. This is true via the Fundamental Theorem of Algebra.