Student: Raine Ma
Username: 433mea
ID#: 43191

Solution is shown below:

3	1	7	6	2	$^{\text{R}}4$	5
4	7	2	5	6	1	3
7	3	5	2	1	6	$^{\text{A}}4$
1	2	6	4	$^{\text{R}}5$	3	7
2	$^{\text{R}}5$	1	3	4	7	6
6	4	3	1	$^{\text{A}}7$	5	2
$^{\text{A}}5$	6	4	7	3	2	1

Examining the problem, we notice that it would likely take very complex casework and proofs to solve this problem. Also, I could potentially leave out many cases due to human error. However, since use of computer programs is allowed on this contest, I wrote a Python program to exhaustively search all possible grids. The program utilizes backtracking, which is a well-known algorithm used in constraint satisfaction problems such as this one.

```python
from time import time

start = time()

"""
I used
Python 3.11.2 (v3.11.2:878ead1ac1, Feb  7 2023, 10:02:41)
[Clang 13.0.0 (clang-1300.0.29.30)] on darwin
"""

def isMagic(grid, target):
    #Checks whether grid is magic square
    lis = [sum(grid[0:3]),sum(grid[3:6]),sum(grid[6:9]),
           sum(grid[::3]),sum(grid[1::3]),sum(grid[2::3]),
           sum(grid[::4]),grid[2]+grid[4]+grid[6]]
    return len(set(lis)) == 1 and lis[0] == target

def toStr(grid):
    return 'skibidi'.join(map(str, grid))
    #creates unique representation for each grid
    #since lists are unhashable
```

```python

def atMost8(grid):
    return len(set(grid)) < 9
    #Checks whether at most 8 distinct values in grid

def rotate(grid): #Rotate 90 clockwise
    return [grid[6], grid[3], grid[0],
            grid[7], grid[4], grid[1],
            grid[8], grid[5], grid[2]]

def reflect(grid): #switch cols 1 and 3
    return [grid[2], grid[1], grid[0],
            grid[5], grid[4], grid[3],
            grid[8], grid[7], grid[6]]

def syms(grid): #Generates all possible symmetric variations of grid
    s = []
    current = grid
    for i in range(4):
        s.append(current)
        s.append(reflect(current))
        current = rotate(current)
    return s

def search(ind, left, grid, dist_vals, target, sols, seen):
    if ind == len(left): #If grid is filled already
        if isMagic(grid, target):
            #appends solution if it has not been discovered
```

```
            symmetries = syms(grid)
            s = min(toStr(sym) for sym in symmetries)
            #Distinct string representation for each grid
            #Ensures we are not overcounting
            if s not in seen:
                seen.add(s)
                sols.append(grid[:])
        return
    #else, if grid is not filled keep going
    pos = left[ind]
    for val in range(1, 13): #Iterates 1-12
        if val not in dist_vals or len(dist_vals) < 8:
        #Checking whether value is distinct in grid
            #Whether adding another distinct value would violate at most 8
            grid[pos] = val #If it can be added set current pos to val
            new = val not in dist_vals #If val is not previously used in grid
            if new: #Add to distinct values tracker if new
                dist_vals.add(val)

            row, col = pos // 3, pos % 3
            #bools that check whethers any constraints set by problem are violated
            #this check before helps reduce the amount of times the code runs
            #will save some exexcution time
            bool1 = (all(grid[3 * row + c] != 0 for c in range(3)) and
                     sum(grid[3 * row + c] for c in range(3)) != target)
            bool2 = (all(grid[r * 3 + col] != 0 for r in range(3))
                     and sum(grid[r * 3 + col] for r in range(3)) != target)
            #Checks rows and columns
```

```
            bool3 = (pos in [0, 4, 8] and all(grid[i] != 0 for i in [0, 4, 8])
                     and sum(grid[i] for i in [0, 4, 8]) != target)
            bool4 = (pos in [2, 4, 6] and all(grid[i] != 0 for i in [2, 4, 6]) and
                     sum(grid[i] for i in [2, 4, 6]) != target)
            #checks diagonals
            valid = not bool1 and not bool2 and not bool3 and not bool4
            #whether grid satisfies conditions

            if valid: #Moves on to next position if grid is valid
                search(ind + 1, left, grid, dist_vals, target, sols, seen)

            if new: #Remove if it is new
                dist_vals.remove(val)
            grid[pos] = 0 #reset for next iteration

sols = []
seen = set()

for target in range(3,37):
    """
    Sum of integers 1-12 = 108, so strong upper bound
    on sums not exceeding 108/3 = 36.
    If grid contains all 1s, sum is 9, so strong lower
    bound of sum being greater than or equal to 3.
    """
    for center in range(1, 13):
        #Grid can contain ints 1-12, center can be 1-12
        grid = [0] * 9
```

```python
        grid[4] = center #set center

        blanks = [i for i in range(9)]
        #Iterate based on positions of 1 and 2
        #assign 1 and 2 to distinct positions
        for i in range(len(blanks)):
            for j in range(i + 1, len(blanks)):
                for pair in [(1, 2), (2, 1)]:
                    temp = grid[:]
                    temp[blanks[i]] = pair[0]#Check all possible positions of 1 and 2
                    temp[blanks[j]] = pair[1]
                    #initialize positions left
                    left = [p for p in blanks if p not in [blanks[i], blanks[j]]]
                    dist_vals = {1, 2, center}
                    #tracks how many distinct vals left
                    #begin backtracking from index 0
                    search(0, left, temp, dist_vals, target, sols, seen)

for sol in sols:
    if not atMost8(sol):
        sols = list(filter((sol).__ne__, sols))
        #Removes sols with more than 8 distinct values

for sol in sols:
    print(sol) #prints distinct solutions

print("The program took " + str(time()-start) + " seconds") #runtime check (for fun)
```

Note that the program exhaustively tries every single possible grid, calculating all 5 sums for each. The program also ensures no grid has its rotations and reflections overcounted. It also filters through solutions to ensure they only have at most eight distinct values. Therefore, this program has searched through all possible grids. The output is shown below:

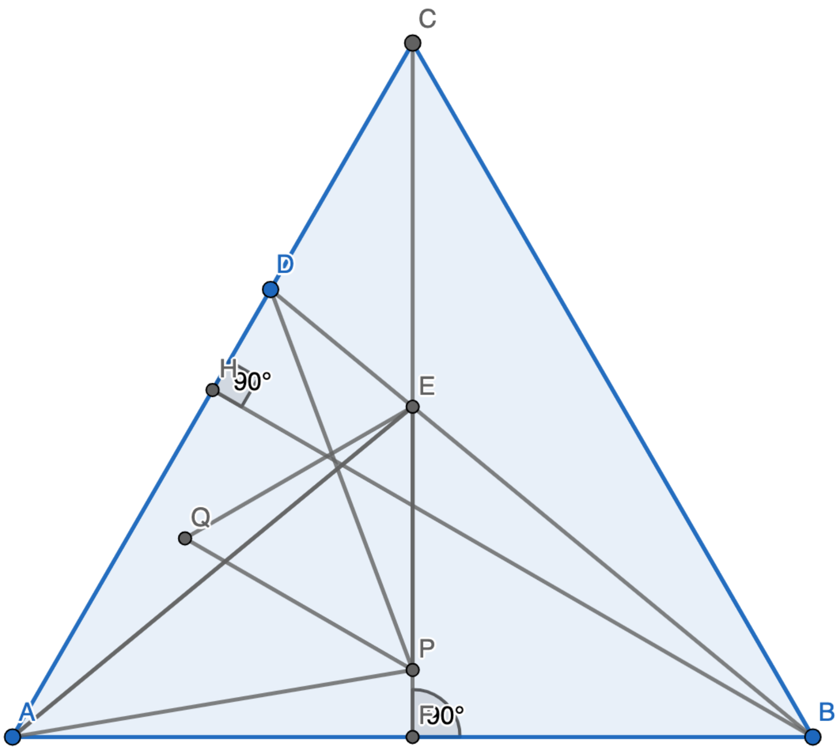
```
[2, 1, 3, 3, 2, 1, 1, 3, 2]
[2, 3, 4, 5, 3, 1, 2, 3, 4]
[2, 5, 5, 7, 4, 1, 3, 3, 6]
The program took 104.74400997161865 seconds
```

Therefore, there are $\boxed{3}$ distinct solutions, namely:

$$\begin{bmatrix} 2 & 1 & 3 \\ 3 & 2 & 1 \\ 1 & 3 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 3 & 4 \\ 5 & 3 & 1 \\ 2 & 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 5 & 5 \\ 7 & 4 & 1 \\ 3 & 3 & 6 \end{bmatrix}$$



We first prove "If $\overline{AB}$ is perpendicular to $\overline{CE}$, then $\triangle EPQ$ is equilateral." Choose D arbitrarily.

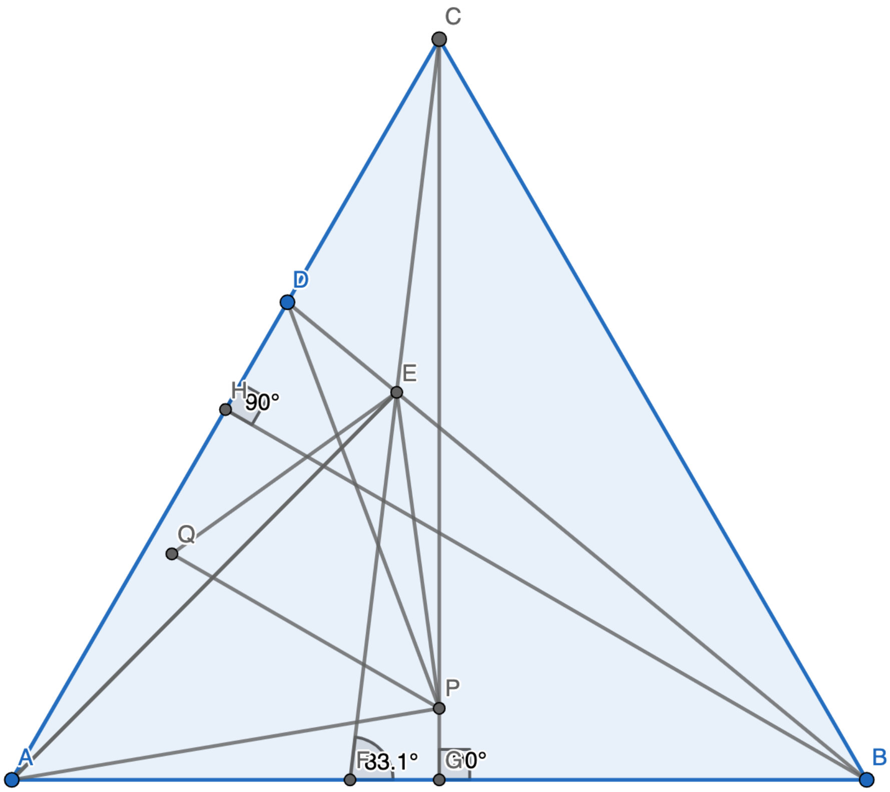
First, let us begin angle chasing. Let $m\angle DBA = \theta$. By definition of a circumcircle, $m\angle DPA = 2\theta$. By triangle angle sum, $m\angle BEP = 90 - \theta$. $\overline{\text{AE}} = \overline{\text{EB}}$, and $m\angle AFE = m\angle BEF = 90°$, so $m\angle AEP = m\angle BEP = 90 - \theta$ by the perpendicular bisector theorem and isosceles triangle theorem. $m\angle AEP + m < DEA = 90 + \theta$, therefore $m\angle DEA = 2\theta$. Notice

$$m\angle DEA = m\angle DPA$$

Therefore, $ADEP$ is cyclic. Since $\triangle DEA$ is inside $ADEP$, it follows that the center of $ADEP$ is Q, since Q is the circumcenter of $\triangle DEA$. Since P and E both lie on a circle with center Q, $\overline{\mathrm{PQ}} = \overline{\mathrm{QE}}$. By the isosceles triangle theorem, $m\angle QPE = m\angle QEP$.

Observe that both P and Q are equidistant from both A and D by definition of a circumcenter. By the converse of the perpendicular bisector theorem, both P and Q must lie on the perpendicular bisector of $\overline{\mathrm{AD}}$. Note that the perpendicular bisector of $\overline{\mathrm{AD}}$ is parallel to $\overleftrightarrow{BH}$, since they both form $90°$ angles with $\overleftrightarrow{AC}$. Therefore, $\overleftrightarrow{PQ} \parallel \overleftrightarrow{BH}$.

P lies on $\overline{CF}$, as it must be equidistant from both A and B. Since $CE \perp AB$, E lies on CF. Therefore, $\overleftrightarrow{PE} \parallel \overleftrightarrow{CF}$. Since $\overleftrightarrow{BH}$ makes a $60°$ angle with $\overleftrightarrow{CF}$ it follows that $\overleftrightarrow{PQ}$ also makes a $60°$ angle with $\overleftrightarrow{PE}$. By this, $m\angle QPE = 60°$. Therefore, $m\angle QPE = m\angle QEP = 60°$. By triangle angle sum, $m\angle QPE = m\angle QEP = m\angle EQP = 60°$. Therefore, $\triangle EPQ$ is equilateral.



Now we prove: "If $\overleftrightarrow{AB}$ is not perpendicular to $\overleftrightarrow{CE}$, then $\triangle EPQ$ is not equilateral."

Consider $\angle QPE$. We have proved that $\overleftrightarrow{PQ} \parallel \overleftrightarrow{BH}$, by the definition of a circumcenter. For $\overleftrightarrow{PE}$ to make a $60°$ angle with $\overleftrightarrow{QP}$, it must be that $\overleftrightarrow{PE} \parallel \overleftrightarrow{\mathrm{CG}}$. P lies on $\overleftrightarrow{CG}$ by definition of a circumcenter. Therefore, $\overleftrightarrow{PE} \parallel \overleftrightarrow{CG}$ if and only if E lies on $\overline{\mathrm{CG}}$. If E was on $\overleftrightarrow{CG}$, $\overline{CE} \perp \overline{AB}$, violating our assumption that $\overline{AB} \not\perp \overline{CE}$. Since $PE \nparallel CG$, this means that $\overleftrightarrow{PE}$ will form some angle other than $60°$ with $\overleftrightarrow{PQ}$. Therefore, $m\angle QPE \neq 60°$ in this case. Therefore, if $\overline{\mathrm{AB}} \not\perp \overline{\mathrm{CD}}$, $\triangle EPQ$ is not an equilateral triangle. We conclude that $\triangle EPQ$ is equilateral triangle if and only if $\overline{\mathrm{AB}} \perp \overline{\mathrm{CD}}$.

Observe that for x_i to occupy position k in T, exactly $k-1$ elements from $x_1, x_2, \ldots, x_{i-1}$ must precede x_k. There are $\binom{i-1}{k-1}$ ways to select these elements. For the other $n-i$ elements, they do not affect the position, but we must consider them. Hence, the probability that x_i occupies position k is

$$\frac{\binom{i-1}{k-1} \cdot 2^{n-i}}{2^n} = \frac{\binom{i-1}{k-1}}{2^i}$$

This holds for all $i \neq 1$, we will consider this case later. Denote $E[x_i]$ the expected contribution of x_i, where $i \neq 1$. Then

$$E[x_i] = \sum_{k=1}^{i} (-1)^{k+1} \cdot k \cdot x_i \cdot \frac{\binom{i-1}{k-1}}{2^i}$$

Define the inner sum

$$S_i = \sum_{k=1}^{i} (-1)^{k+1} \cdot k \cdot \binom{i-1}{k-1}$$

Then

$$E[x_i] = \frac{x_i}{2^i} \cdot S_i$$

We can rewrite S_i as

$$S_i = \sum_{k=1}^{i} (-1)^{k+1} \cdot (i-1) \cdot \binom{i-2}{k-2}$$

Let $j = k-2$. $\binom{i-2}{j} = 0$ for $j < 0$, so the sum effectively starts at $j = 0$.

$$S_i = -(i-1) \sum_{j=0}^{i-2} (-1)^j \cdot \binom{i-2}{j}$$

By the binomial theorem, this equals

$$-(i-1) \cdot (1-1)^{i-2} = -(i-1) \cdot 0^{i-2}$$

This evaluates to -1 for $i = 2$, and 0 for all $i > 2$. Substituting S_i back into $E[x_i]$, we obtain

$$E[x_i] = x_i \cdot \frac{\begin{cases} 0 & x > 2 \\ -1 & x = 2 \end{cases}}{2^i}$$

Using this formula, we see that x_2 has an expected contribution of $-\frac{x_2}{4}$ and all x_i where $i > 2$ have a contribution of zero. We now consider x_1. Note that x_1 will appear in exactly half of the subsets, and when it does, it will always be the first element. Therefore, the expected contribution of x_1 is $\frac{x_1}{2}$. Summing all cases, the expected value of T is

$$\boxed{\frac{x_1}{2} - \frac{x_2}{4}}$$

.

Student: Raine Ma
Username: 433mea
ID#: 43191

Define

$$\alpha = \sqrt[3]{5} + \sqrt[3]{25}$$

Observe that the following equation holds:

$$\alpha^3 = 15\alpha + 30$$

We prove that any powers of α greater than or equal to 3 can be expressed in terms of $1, \alpha$, and α^2. We proceed with induction. Assume that

$$z \cdot \alpha^k = b_0 + b_1\alpha + b_2\alpha^2$$

for some $k \geq 0$, where $z, b_0, b_1, b_2 \in \mathbb{Z}$. Then

$$z \cdot \alpha^{k+1} = b_0\alpha + b_1\alpha^2 + b_2\alpha^3$$

Substitute $\alpha^3 = 15\alpha + 30$ and simplify to obtain

$$z \cdot \alpha^{k+1} = 30b_2 + (b_0 + 15b_2)\alpha + b_1\alpha^2$$

Therefore, if $z \cdot a^k$ can be expressed as $b_0 + b_1\alpha + b_2\alpha^2$, where $z, b_0, b_1, b_2, \in \mathbb{Z}$, then $z \cdot a^{k+1}$ can as well. Since the base case $k = 0$ holds, this result holds for all z, k we are interested in.

We now prove that $2\sqrt[3]{5} + 3\sqrt[3]{25}$ cannot be expressed in terms of α^0, α^1, or α^2 with integer coefficients. For the sake of contradiction, assume that some $c_0, c_1, c_2 \in \mathbb{Z}$ satisfy

$$2\sqrt[3]{5} + 3\sqrt[3]{25} = c_0 + c_1\alpha + c_2\alpha^2$$

Substituting $\alpha = \sqrt[3]{5} + \sqrt[3]{25}$, we obtain

$$c_0 + c_1\alpha + c_2\alpha^2 = c_0 + c_1\sqrt[3]{5} + c_1\sqrt[3]{25} + c_2\sqrt[3]{25} + 10c_2 + 5c_2\sqrt[3]{5}$$

Group like terms:

$$(c_0 + 10c_2) + (c_1 + 5c_2)\sqrt[3]{5} + (c_1 + c_2)\sqrt[3]{25} = 2\sqrt[3]{5} + 3\sqrt[3]{25}$$

We obtain a system of equations

$$\begin{cases} (1) \quad c_0 + 10c_2 = 0 \\ (2) \quad c_1 + 5c_2 = 2 \\ (3) \quad c_1 + c_2 = 3 \end{cases}$$

From equation (3):

$$c_1 + c_2 = 3 \implies c_1 = 3 - c_2$$

Substitute this result into equation (2):

$$3 + 4c_2 = 2$$

We find that $c_2 = -\frac{1}{4}$, which is not an integer. This violates our assumption that c_0, c_1, and c_2 are integers. Therefore, $2\sqrt[3]{5} + 3\sqrt[3]{25}$ cannot be expressed in terms of $c_0 + c_1\alpha + c_2\alpha^2$, where $c_0, c_1, c_2 \in \mathbb{Z}$. Our previous result found that any $z \cdot \alpha^k$ can be expressed in terms of $1, \alpha, \alpha^2$, and therefore any sum of $z \cdot \alpha^k$, where $z \in \mathbb{Z}$ and $k \in \mathbb{N}$. Therefore, $2\sqrt[3]{5} + 3\sqrt[3]{25}$ cannot be expressed as a sum of integer multiples of powers of α, meaning that no polynomial with integer coefficients exists such that $P(\alpha) = 2\sqrt[3]{5} + 3\sqrt[3]{25}$.