Analysis: Indicium

Test Set 1

There are a few different options for solving test set 1. Since there are only 44 possible cases, one option is to generate all answers by hand or via a program that is run locally, then submit a program that dispenses those. Another approach is to notice that there are not many different Latin squares for **N** ≤ 5 (see the number of Latin squares [here](#)), and check them all. To generate all Latin squares, we can recursively fill in the cells one by one. For each cell, we try all **N** possible values. For each one, we ensure that it does not conflict with any cells in the same row or same column. Since there are at most 161280 Latin squares to consider, this is quite quick.

Test Set 2

Unfortunately, once **N** gets even slightly large, there are way too many Latin squares to generate them all (for **N** = 11, for example, there are 776966836171770144107444346734230682311065600000 different Latin squares).

There are many creative ways to solve this test set. The [Code Jam forum](#) is a good place to share and discuss different solutions! For example, we can directly create Latin squares with the appropriate trace by modifying structured Latin squares (for example, by modifying [circulant Latin squares](#)). Below, we discuss an easy-to-implement idea which is a little tricky to come up with and uses a graph algorithm in the middle!

First, we start by dealing with the impossible cases. If **K** = **N**+1, then the only possible diagonals have exactly one `2` and **N**-1 `1`s. However, if **N**-1 of the diagonal elements are `1`, then the only location for the `1` in the remaining row must be on the diagonal, so we cannot make a sum of **N**+1. Similarly, we cannot make a sum of N^2-1 since the only possible diagonal is one `N-1` and **N**-1 `N`s.

We will now show a construction which works for every other case (with 2 additional small cases that don't work, see below). One of the main insights needed is that all possible sums are achievable using a diagonal with almost all values the same. In particular, we may assume that at least **N**-2 values are the same: AAAA ... AABC for some A, B, C (not necessarily all different).

For example, if **N** = 10 and **K** = 20, we can choose A = 2, B = 2, and C = 2. If **N** = 10 and **K** = 55, we can choose A = 6, B = 4, and C = 3. We already showed above that A = B if and only if A = C. We leave it as an exercise to show that all values for **K** between **N** and N^2 are possible with these constraints. (Note: you have to be a little careful with **N** = 3. If B = C, then A = B = C for a similar reason; so with **N** = 3, neither **K** = 5 nor **K** = 7 will have solutions). To find the appropriate values of A, B, and C, we can brute force all possible triples and check whether the chosen diagonal will work.

Now that we know what the diagonal looks like, how do we actually find a Latin square that has this diagonal? To do that, we will fill in the unfilled cells row by row. We will use [bipartite matching](#) to find a valid row. In one bipartition, we have **N** vertices for the **N** cells in that row. In the other bipartition, we have **N** vertices for the **N** numbers that can be placed into the cell. Make an edge between the cell vertex on the diagonal and the number vertex that was decided on. For every other cell, make an edge between a cell vertex and a number vertex if that number can be put into that cell without breaking the Latin square properties.

We can greedily pick any perfect matching for each row *starting with the rows with B and C on their diagonal*. Once we have filled in these two rows, we can use [Hall's Marriage Theorem](#) to show that we will never run into any issues (so long as the conditions above about A, B, C are met).

Hall's Theorem

This section is dedicated to proving the above claim that Hall's theorem holds. We will assume in this section that the reader is comfortable with Hall's theorem. A one sentence high-level reminder of Hall's theorem: *All subsets of one bipartition have a neighborhood that is at least as large as the original subset if and only if the graph has a perfect matching.*

For the explanation here, we will make the top two rows with B and C on their diagonal as the top two rows. We'll assume that these two rows are already filled in (and leave the proof you can do this to the reader). The important part is that the top-left 2 × 2 submatrix is `CA`/`AB`. Now imagine that we have filled in **N**-k rows (and have k mostly empty rows). Consider this example with **N** = 8 and k = 3. (`?` means filled in, but it doesn't matter with what and `_` means not filled in yet):

```
CA??????
AB??????
??A?????
???A????
????A???
_____A__
_____A_
_____A
```

For each of the **N**-1 non-`A` "cell vertices", the **N**-k vertices on the left of the diagonal have a degree of k and the k-1 vertices on the right of the diagonal have a degree of k-1 (because the number `A` is also restricted). For each of the **N**-1 non-`A` "number vertices", each number originally had degree **N** and we have removed at least **N**-k of those edges since the number appeared once in the top **N**-k rows. Thus, the maximum degree of the "number vertices" is k.

We will ignore the "cell vertex" and the "number vertex" corresponding to the forced diagonal entry since that will be forced in our matching (and leaving it out makes our math below easier).

Let X be a subset of "cell vertices". Let m = |X|. We must show that |N(X)| ≥ m in order to utilize Hall's theorem (where N(X) is the set of "number vertices" that are adjacent to at least one vertex in X). We have 2 separate cases:

> **Case 1:** m ≤ k-1.
>
> Since the degree of each vertex in X is at least k-1, the number of edges leaving X is at least m × (k-1). Consider the "number vertices" that these edges are absorbed into. Since the maximum degree of "number vertices" is k, there are at least (m × (k-1))/k "number vertices" that absorb these edges. That is, |N(X)| ≥ (m × (k-1))/k = m-m/k. Since m ≤ k-1, we have that m/k < 1. So |N(X)| > m-1. Since |N(X)| is an integer, we have |N(X)| ≥ m as desired.
>
> **Case 2:** m ≥ k.
>
> Consider the edges leaving X. At most k-1 of them have degree k-1, and the remaining have degree k. Thus, the number of edges leaving X is at least (k-1) × (k-1) + (m-(k-1)) × k. Since the maximum degree of "number vertices" is k, there are at least ((k-1) × (k-1) + (m-(k-1)) × k)/k "number vertices" that absorb these edges. That is, |N(X)| ≥ ((k-1) × (k-1) + (m-

(k-1)) × k)/k = m - (1 - 1/k). Since 1 - 1/k < 1, we have |N(X)| > m-1. Since |N(X)| is an integer, we have |N(X)| ≥ m as desired.

Thus, in all cases, the conditions for Hall's theorem are satisfied, so there exists a perfect matching and we can iteratively complete the Latin square.