**Part 4** : "New Advanced Study of Magic Squares and Cubes"

**Chapter 4** : Commentary Articles No.2 by **Kanji Setsuda**:
"Various Arts and Tools for Studying Magic Squares"

# Section 12 : What does "New Euler's Method" Mean? What else can we Apply it for?

## 0. Our Purpose

I would like to do something new this time. How about discussing what our "New Euler's Method" means in total and what else it can be applied for?

In the previous sections we have examined our new method by applying it to almost every typical magic square of orders 3, 4, 5, 7, 8 and 9 just to know what a wide range of them it could compose successfully.

Indeed we could make almost everything important of low orders, and build various interesting magic squares of high orders, really impressive, rare and precious jewels.

Although our "New Euler's Method" was just a hypothesis at first, it could pass various tests, demonstrate its power enough to solve any problem, and show its own meaning.

But why is it so powerful? What makes it so effective, can you guess?

## 1. What makes New Euler's Method so Powerful?

(1) I consider that its notation must be one of the reasons. We always use Positional Number System of Base 'N' in our new method. We made Euler Squares of order 3, 5 and 7 with PNS of base 3, 5 and 7. We built Complete Euler Squares of order 4 and 8 with PNS of base 2, that means Binary System, and of order 9 with base 3.

The power of New Euler's Method may probably come out of this notation itself. Let me explain that by the next examples of order 5 and 7.

The first two diagrams show the 2-dimensional coordinates of each position of 5x5 array on the left hand side and on the right the series of integers 0~24 put in order there and written in the notation of 5th increment.

**** Basic Diagrams for Magic Squares 5x5 ****

[Coordinates of its Position]

| | | | | |
|---|---|---|---|---|
| (0,0) | (0,1) | (0,2) | (0,3) | (0,4) |
| (1,0) | (1,1) | (1,2) | (1,3) | (1,4) |
| (2,0) | (2,1) | (2,2) | (2,3) | (2,4) |
| (3,0) | (3,1) | (3,2) | (3,3) | (3,4) |
| (4,0) | (4,1) | (4,2) | (4,3) | (4,4) |

[Values by /N5i]

| | | | | |
|---|---|---|---|---|
| 00 | 01 | 02 | 03 | 04 |
| 10 | 11 | 12 | 13 | 14 |
| 20 | 21 | 22 | 23 | 24 |
| 30 | 31 | 32 | 33 | 34 |
| 40 | 41 | 42 | 43 | 44 |

| [Modern Notation in Decimal] | [PNS of Base 5] /N5i | [Decompositions by Base 5] /D5i H/ | L/ |
|---|---|---|---|
| 0 1 2 3 4 | 00 01 02 03 04 | 0 0 0 0 0 | 0 1 2 3 4 |
| 5 6 7 8 9 | 10 11 12 13 14 | 1 1 1 1 1 | 0 1 2 3 4 |
| 10 11 12 13 14 | 20 21 22 23 24 | 2 2 2 2 2 | 0 1 2 3 4 |
| 15 16 17 18 19 | 30 31 32 33 34 | 3 3 3 3 3 | 0 1 2 3 4 |
| 20 21 22 23 24 | 40 41 42 43 44 | 4 4 4 4 4 | 0 1 2 3 4 |

As you see, both the coordinates and values (/N5i) are expressed in the very similar form of numerical expression. It is really effective for them to have these similar expressions in common.

```
** Basic Diagrams for Magic Squares 7x7 **
  [PNS of Base 7]      [Decompositions by Base 7]
/N7i                 /D7i    High/          Low/
00 01 02 03 04 05 06  0 0 0 0 0 0 0  0 1 2 3 4 5 6
10 11 12 13 14 15 16  1 1 1 1 1 1 1  0 1 2 3 4 5 6
20 21 22 23 24 25 26  2 2 2 2 2 2 2  0 1 2 3 4 5 6
30 31 32 33 34 35 36  3 3 3 3 3 3 3  0 1 2 3 4 5 6
40 41 42 43 44 45 46  4 4 4 4 4 4 4  0 1 2 3 4 5 6
50 51 52 53 54 55 56  5 5 5 5 5 5 5  0 1 2 3 4 5 6
60 61 62 63 64 65 66  6 6 6 6 6 6 6  0 1 2 3 4 5 6
```

It can surely help them express some unique structures of magic square world, such as the cyclic structure of pan-diagonals. It is because this notation could be considered to be the modulo-remainder function when divided by 'N'.

And the unique transformation "DAMT" and "Prototype Squares" often exchange their coordinate particulars with their value ones themselves.

Decomposition diagrams of order 5 and 7 above are made from the value notation expressed by PNS of Base 5 and 7. Each value is divided into high and low parts, and put separately into two layers. Between the original square and these decomposed layers, the following equations are always active.

$V_5n = 5*Hn + Ln$ (n=1,2,3,4, ... ,24,25) ... MS55

$V_7n = 7*Hn + Ln$ (n=1,2,3,4, ... ,48,49) ... MS77

If you want to have your solution in classical style, you must always add one to each value of every position.

These layers can express very well such a beautiful structure of "Euler Squares" as you have seen many times.

Take your careful look again at those decomposed layers of order 5 and 7 above, and find some interesting properties both of them have in common.

[1] Every pan-diagonal in each layer is made up of {0, 1, 2, 3, 4} or {0, 1, 2, 3, 4, 5, 6}.
Yes. This is just the same as "Complete Euler Squares."

[2] Every column in high layer is also made of {0, 1, 2, 3, 4} or {0, 1, 2, 3, 4, 5, 6}, and every row in low layer is also composed of {0, 1, 2, 3, 4} or {0, 1, 2, 3, 4, 5, 6}.

They are originally the 5x5 and 7x7 arrays. They are also the first representative forms of 'Prototype Squares of order 5 and 7. Though they are only the examples whose coordinates and values are expressed in the same manner, they are already almost "Complete Euler Squares". It seems that you could nearly get the objects just by some simple exchanges of their parts.

(2) Our New Euler's Method is the hypothetical way of composition by which we can expect to build 'Euler Squares' and 'Complete Euler Squares'.

At first we try to make some decomposition layers, and when we get them, we calculate and compose the object solutions.

We assume every layer of decompositions must accept all the definitions of 'Euler Squares' such as:

**Def[1]** Each row, each column (and each pan-diagonal of 'Complete Euler Squares') must be made of {0, 1, 2, 3, 4} or {0, 1, 2, 3, 4. 5, 6} using any number strictly once. Neither repeating employment of any number nor drop-off must be found.

**Def[2]** Each layer must consist of {0, 1, 2, 3, 4} or {0, 1, 2, 3, 4. 5, 6} using any number 5 or 7 times as often as others.

**Def[3]** Every combination of corresponding numbers in all layers must be any one of {00, 01, 02, 03, 04, 10, 11, 12, 13, 14, 20, 21, ... , 42, 43, 44(N5i)} or {00, 01, 02, 03, 04, 05, 06, 10, 11, 12, 13, 14, 15, 16, 20, 21, ... , 62, 63, 64. 65, 66(N7i)} and it must be different from the others. Neither repetition nor drop-off must be found.

On top of that we presume each layer must have the same properties as the total object squares have, such as 'Complete Conditions', 'Self-Complementary Conditions', 'Composite Conditions', and so on, if necessary.

It may sound too strict. Yes, indeed. It is really too strict for you to make very many 'Complete Euler Squares'. You can only make rare, precious jewels, especially in the case of higher orders.

If you want to make any 'Non-Euler Square', you don't have to accept such the very strict conditions. You can really find many Non-Euler Squares in higher orders, I say, far more than Euler ones.

But in the case of order 4 and 5 you cannot find any Non-Complete-Euler-Square at all. It seems to be one of the greatest mysteries of our magic square world.

(3) In our New Euler's Method we make 'Latin Units' at first, and when we get them, then we combine any appropriate units and assume them as 'Decomposed Layers' so that we can calculate and compose the object solutions by them.

When we make Latin Units by ourselves at first, we presume each unit can take any position to sit on, and each unit can take any appropriate partner or members to make a team with for 'Decomposed Layers'. We presuppose each unit is born free and equal in their rights, I would say.

Of course, there are some definite limitations to each unit in reality. There are 'good combinations' or 'good teams'. Only those which can make correct solutions are chosen for good ones. We must cut off those which make wrong answers against the first definitions and also against the list-forming inequality conditions.

When you combine any two units, you must avoid the next two types that would make wrong answers inevitably: the combination of the same unit with itself and the combination of 'Complementary Units'.

Therefore we have to prepare the reference table for the best combinations so as to know if the temporary pair is good or not before we combine them.

But in the case of odd orders since it is not enough to have a single table of similarity, we must make another reference table for Complementary Units and then collect those two tables and make them into one.

Let me show you some examples of the Reference Tables of Simultaneous Magic Squares 5x5: Self-complementary and Pan-diagonal made by our New Euler's Method.

**** Simultaneous MS55: Both Self-Complementary and Pan-diagonal Made by New Euler's Method ****

[Latin Units]

| 1/ | 2/ | 3/ | 4/ | 5/ | 6/ | 7/ | 8/ |
|---|---|---|---|---|---|---|---|
| 02413 | 02431 | 04123 | 04321 | 12304 | 12340 | 13024 | 13420 |
| 41302 | 43102 | 23041 | 21043 | 30412 | 34012 | 24130 | 20134 |
| 30241 | 10243 | 41230 | 43210 | 41230 | 01234 | 30241 | 34201 |
| 24130 | 24310 | 30412 | 10432 | 23041 | 23401 | 41302 | 01342 |
| 13024 | 31024 | 12304 | 32104 | 04123 | 40123 | 02413 | 42013 |

| 9/ | 10/ | 11/ | 12/ | 13/ | 14/ | 15/ | 16/ |
|---|---|---|---|---|---|---|---|
| 31024 | 31420 | 32104 | 32140 | 40123 | 40321 | 42013 | 42031 |
| 24310 | 20314 | 10432 | 14032 | 23401 | 21403 | 01342 | 03142 |
| 10243 | 14203 | 43210 | 03214 | 01234 | 03214 | 34201 | 14203 |
| 43102 | 03142 | 21043 | 21403 | 34012 | 14032 | 20134 | 20314 |
| 02431 | 42031 | 04321 | 40321 | 12340 | 32140 | 13420 | 31420 |

[Count of Latin Units = 16]

**** Reference Tables for the Best Combination ****

[Similar]

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 |
| 2 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 |
| 3 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 |
| 4 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 |
| 6 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 8 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 |
| 9 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 |
| 10 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 |
| 11 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 |
| 12 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 |
| 13 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 |
| 14 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 |
| 15 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 |
| 16 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 |

[Complementary]

| C | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 |
| 2 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 |
| 3 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 |
| 4 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 |
| 6 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 |
| 8 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 |
| 9 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 |
| 10 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 11 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 |
| 12 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 | 5 | 5 |
| 13 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 |
| 14 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 5 | 5 |
| 15 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 |
| 16 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 |

[Collective]

| SC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 |
| 2 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 |
| 3 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 |
| 4 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 |
| 5 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 |
| 6 | 5 | 5 | 5 | 5 | 15 | 25 | 5 | 5 | 5 | 5 | 25 | 15 | 5 | 5 | 5 | 5 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 15 | 15 | 25 | 5 | 5 | 5 | 5 | 5 | 5 |

```
 8|  5  5  5  5  5  5 15 25 25 15  5  5  5  5  5  5
 9|  5  5  5  5  5  5 15 25 25 15  5  5  5  5  5  5
10|  5  5  5  5  5  5 25 15 15 25  5  5  5  5  5  5
11|  5  5  5  5 15 25  5  5  5  5 25 15  5  5  5  5
12|  5  5  5  5 25 15  5  5  5  5 15 25  5  5  5  5
13|  5  5 15 25  5  5  5  5  5  5  5  5 25 15  5  5
14|  5  5 25 15  5  5  5  5  5  5  5  5 15 25  5  5
15| 15 25  5  5  5  5  5  5  5  5  5  5  5  5 25 15
16| 25 15  5  5  5  5  5  5  5  5  5  5  5  5 15 25
```

[**Compositions:** Used Units// List Number#]

```
   3/H   1/L               1#      3/H   2/L               2#
 04123 02413   1 23 10 12 19    04123 02431   1 23 10 14 17
 23041 41302  15 17  4 21  8    23041 43102  15 19  2 21  8
 41230 30241  24  6 13 20  2    41230 10243  22  6 13 20  4
 30412 24130  18  5 22  9 11    30412 24310  18  5 24  7 11
 12304 13024   7 14 16  3 25    12304 31024   9 12 16  3 25

   3/H   5/L               3#      3/H   6/L               4#
 04123 12304   2 23  9 11 20    04123 12340   2 23  9 15 16
 23041 30412  14 16  5 22  8    23041 34012  14 20  1 22  8
 41230 41230  25  7 13 19  1    41230 01234  21  7 13 19  5
 30412 23041  18  4 21 10 12    30412 23401  18  4 25  6 12
 12304 04123   6 15 17  3 24    12304 40123  10 11 17  3 24

[ 5]    3/ 11/   [ 6]    3/ 12/   [ 7]    3/ 15/   [ 8]    3/ 16/
  4 23  7 11 20     4 23  7 15 16    5 23  6 12 19     5 23  6 14 17
 12 16  5 24  8    12 20  1 24  8    11 17  4 25  8    11 19  2 25  8
 25  9 13 17  1    21  9 13 17  5    24 10 13 16  2    22 10 13 16  4
 18  2 21 10 14    18  2 25  6 14    18  1 22  9 15    18  1 24  7 15
  6 15 19  3 22    10 11 19  3 22     7 14 20  3 21     9 12 20  3 21

[ 9]    4/  1/   [10]    4/  2/   [11]    4/  5/   [12]    4/  6/
  1 23 20 12  9     1 23 20 14  7    2 23 19 11 10     2 23 19 15  6
 15  7  4 21 18    15  9  2 21 18   14  6  5 22 18    14 10  1 22 18
 24 16 13 10  2    22 16 13 10  4   25 17 13  9  1    21 17 13  9  5
  8  5 22 19 11     8  5 24 17 11    8  4 21 20 12     8  4 25 16 12
 17 14  6  3 25    19 12  6  3 25   16 15  7  3 24    20 11  7  3 24

[13]    4/ 11/   [14]    4/ 12/   [15]    4/ 15/   [16]    4/ 16/
  4 23 17 11 10     4 23 17 15  6    5 23 16 12  9     5 23 16 14  7
 12  6  5 24 18    12 10  1 24 18   11  7  4 25 18    11  9  2 25 18
 25 19 13  7  1    21 19 13  7  5   24 20 13  6  2    22 20 13  6  4
  8  2 21 20 14     8  2 25 16 14    8  1 22 19 15     8  1 24 17 15
 16 15  9  3 22    20 11  9  3 22   17 14 10  3 21    19 12 10  3 21

[Count = 16]
```

As you see, the best combination of any two units always counts 5 in those reference tables. Any other count cannot make correct answers at all. I don't yet know exactly what makes 5 in this case, but I feel any unit would refuse to be combined with any other partner that is too similar to or too different from itself. 'Interaction' between any two units needs to be less active or more 'random', I would say in other words, than any other combinations.

7 is the best count for the combination in the case of order 7.

(4) At the final stage we calculate and compose the 'Primitive' answers. We check those results by the two types of test: whether they are against the Definition[3] of Euler Squares, and whether they are against the list-forming inequality conditions.

We think what could pass the tests all right should be the final answers, the complete set of object solutions.

It is amazing that we could surely succeed in getting them every time when we tried to by our new method. We could compose every important 'Complete Euler Square' we have just wanted. We could never get any other type of solutions, and we could never lose any important solutions, either. Our New Euler's Method has always demonstrated to be a complete method of composition, I would say.

We presume that variety of combination of Latin Units must be the basis of variety of object solutions. The latter variety should come out of the former one. Every result seems to show that our presumption is true. How great it is!

But why could it be true?

I believe it is because the list of 'Latin Units' and the combinations of them are the basic, essential 'elements' of 'Euler Squares'.

Yes. I should say, our new method combines the Latin Units and the combinations of them with the complete set of object solutions, and puts the one-to-one correspondence between them. It is made possible by the composition/decomposition with the Positional Number System of Base 'N'.

If you only have the list of Latin Units and the number list of Used Units, you can really recompose all the solutions you want under the next calculation:

$V_5n=5*Hn+Ln+1$ (n=1, 2, 3, 4, ... , 24, 25) [in Classical style]

I am sure it is one of the most wonderful things of all.

## 2. What else can you apply it to?

The next job I would like to think of is the application of this powerful method. Which fields can we apply it for?

Let's begin our discussion with a simple little thing.

**(1) For Data Save**: Because you can recompose all the solution list afterward, if you only have the list of Latin Units and the number list of Used Units, you don't have to save all the solution list to your outer memory devices such as any disk or papers.

You may only record the list of Latin Units with the number list of Used Units in place of the whole solution list you have wanted.

That may surely save your natural resources, such as your memory, paper, time or energy, don't you think?

```
 ** Pan-diagonal Magic Squares 4x4 Made by New Euler's Method **
[Latin Units]
       1/        2/        3/        4/
   0 1 1 0   0 1 0 1   0 1 0 1   0 0 1 1
   1 0 0 1   1 0 1 0   0 1 0 1   1 1 0 0
   0 1 1 0   1 0 1 0   1 0 1 0   0 0 1 1
   1 0 0 1   0 1 0 1   1 0 1 0   1 1 0 0
       5/        6/        7/        8/
   1 1 0 0   1 0 1 0   1 0 1 0   1 0 0 1
   0 0 1 1   1 0 1 0   0 1 0 1   0 1 1 0
   1 1 0 0   0 1 0 1   0 1 0 1   1 0 0 1
   0 0 1 1   0 1 0 1   1 0 1 0   0 1 1 0    [Count =  8]
```

[**Reference Table** for the Best Combinations]

| * | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 16 | 8 | 8 | 8 | 8 | 8 | 8 | 0 |
| 2 | 8 | 16 | 8 | 8 | 8 | 8 | 0 | 8 |
| 3 | 8 | 8 | 16 | 8 | 8 | 0 | 8 | 8 |
| 4 | 8 | 8 | 8 | 16 | 0 | 8 | 8 | 8 |
| 5 | 8 | 8 | 8 | 0 | 16 | 8 | 8 | 8 |
| 6 | 8 | 8 | 0 | 8 | 8 | 16 | 8 | 8 |
| 7 | 8 | 0 | 8 | 8 | 8 | 8 | 16 | 8 |
| 8 | 0 | 8 | 8 | 8 | 8 | 8 | 8 | 16 |

**** List of Compositions: Used Units// List Number# ****

| 1/ | 2/ | 3/ | 4/ | 1# |
|---|---|---|---|---|
| 0 1 1 0 | 0 1 0 1 | 0 1 0 1 | 0 0 1 1 | 1 15 10 8 |
| 1 0 0 1 | 1 0 1 0 | 0 1 0 1 | 1 1 0 0 | 14 4 5 11 |
| 0 1 1 0 | 1 0 1 0 | 1 0 1 0 | 0 0 1 1 | 7 9 16 2 |
| 1 0 0 1 | 0 1 0 1 | 1 0 1 0 | 1 1 0 0 | 12 6 3 13 |

| 1/ | 2/ | 3/ | 5/ | 2# |
|---|---|---|---|---|
| 0 1 1 0 | 0 1 0 1 | 0 1 0 1 | 1 1 0 0 | 2 16 9 7 |
| 1 0 0 1 | 1 0 1 0 | 0 1 0 1 | 0 0 1 1 | 13 3 6 12 |
| 0 1 1 0 | 1 0 1 0 | 1 0 1 0 | 1 1 0 0 | 8 10 15 1 |
| 1 0 0 1 | 0 1 0 1 | 1 0 1 0 | 0 0 1 1 | 11 5 4 14 |

| 1/ | 2/ | 4/ | 3/ | 3# |
|---|---|---|---|---|
| 0 1 1 0 | 0 1 0 1 | 0 0 1 1 | 0 1 0 1 | 1 14 11 8 |
| 1 0 0 1 | 1 0 1 0 | 1 1 0 0 | 0 1 0 1 | 15 4 5 10 |
| 0 1 1 0 | 1 0 1 0 | 0 0 1 1 | 1 0 1 0 | 6 9 16 3 |
| 1 0 0 1 | 0 1 0 1 | 1 1 0 0 | 1 0 1 0 | 12 7 2 13 |

| 1/ | 2/ | 4/ | 6/ | 4# |
|---|---|---|---|---|
| 0 1 1 0 | 0 1 0 1 | 0 0 1 1 | 1 0 1 0 | 2 13 12 7 |
| 1 0 0 1 | 1 0 1 0 | 1 1 0 0 | 1 0 1 0 | 16 3 6 9 |
| 0 1 1 0 | 1 0 1 0 | 0 0 1 1 | 0 1 0 1 | 5 10 15 4 |
| 1 0 0 1 | 0 1 0 1 | 1 1 0 0 | 0 1 0 1 | 11 8 1 14 |

| 5# 1/2/5/3/ | 6# 1/2/5/6/ | 7# 1/2/6/4/ | 8# 1/2/6/5/ |
|---|---|---|---|
| 3 16 9 6 | 4 15 10 5 | 3 13 12 6 | 4 14 11 5 |
| 13 2 7 12 | 14 1 8 11 | 16 2 7 9 | 15 1 8 10 |
| 8 11 14 1 | 7 12 13 2 | 5 11 14 4 | 6 12 13 3 |
| 10 5 4 15 | 9 6 3 16 | 10 8 1 15 | 9 7 2 16 |

| 9# 1/3/2/4/ | 10# 1/3/2/5/ | 11# 1/3/4/2/ | 12# 1/3/4/7/ |
|---|---|---|---|
| 1 15 10 8 | 2 16 9 7 | 1 14 11 8 | 2 13 12 7 |
| 12 6 3 13 | 11 5 4 14 | 12 7 2 13 | 11 8 1 14 |
| 7 9 16 2 | 8 10 15 1 | 6 9 16 3 | 5 10 15 4 |
| 14 4 5 11 | 13 3 6 12 | 15 4 5 10 | 16 3 6 9 |

| 13# 1/3/5/2/ | 14# 1/3/5/7/ | 15# 1/3/7/4/ | 16# 1/3/7/5/ |
|---|---|---|---|
| 3 16 9 6 | 4 15 10 5 | 3 13 12 6 | 4 14 11 5 |
| 10 5 4 15 | 9 6 3 16 | 10 8 1 15 | 9 7 2 16 |
| 8 11 14 1 | 7 12 13 2 | 5 11 14 4 | 6 12 13 3 |
| 13 2 7 12 | 14 1 8 11 | 16 2 7 9 | 15 1 8 10 |

```
  17# 1/4/2/3/   18# 1/4/2/6/   19# 1/4/3/2/   20# 1/4/3/7/
    1 12 13  8     2 11 14  7     1 12 13  8     2 11 14  7
   15  6  3 10    16  5  4  9    14  7  2 11    13  8  1 12
    4  9 16  5     3 10 15  6     4  9 16  5     3 10 15  6
   14  7  2 11    13  8  1 12    15  6  3 10    16  5  4  9

  21# 1/4/6/2/   22# 1/4/6/7/   23# 1/4/7/3/   24# 1/4/7/6/
    3 10 15  6     4  9 16  5     3 10 15  6     4  9 16  5
   16  5  4  9    15  6  3 10    13  8  1 12    14  7  2 11
    2 11 14  7     1 12 13  8     2 11 14  7     1 12 13  8
   13  8  1 12    14  7  2 11    16  5  4  9    15  6  3 10

  25# 3/1/2/4/   26# 3/1/2/5/   27# 3/1/4/2/   28# 3/1/4/7/
    1 15  6 12     2 16  5 11     1 14  7 12     2 13  8 11
    8 10  3 13     7  9  4 14     8 11  2 13     7 12  1 14
   11  5 16  2    12  6 15  1    10  5 16  3     9  6 15  4
   14  4  9  7    13  3 10  8    15  4  9  6    16  3 10  5

  29# 3/1/5/2/   30# 3/1/5/7/   31# 3/1/7/4/   32# 3/1/7/5/
    3 16  5 10     4 15  6  9     3 13  8 10     4 14  7  9
    6  9  4 15     5 10  3 16     6 12  1 15     5 11  2 16
   12  7 14  1    11  8 13  2     9  7 14  4    10  8 13  3
   13  2 11  8    14  1 12  7    16  2 11  5    15  1 12  6

  33# 3/4/1/2/   34# 3/4/1/7/   35# 3/5/1/2/   36# 3/5/1/7/
    1 12  7 14     2 11  8 13     5 16  3 10     6 15  4  9
    8 13  2 11     7 14  1 12     4  9  6 15     3 10  5 16
   10  3 16  5     9  4 15  6    14  7 12  1    13  8 11  2
   15  6  9  4    16  5 10  3    11  2 13  8    12  1 14  7

  37# 4/1/2/3/   38# 4/1/2/6/   39# 4/1/3/2/   40# 4/1/3/7/
    1  8 13 12     2  7 14 11     1  8 13 12     2  7 14 11
   15 10  3  6    16  9  4  5    14 11  2  7    13 12  1  8
    4  5 16  9     3  6 15 10     4  5 16  9     3  6 15 10
   14 11  2  7    13 12  1  8    15 10  3  6    16  9  4  5

  41# 4/1/6/2/   42# 4/1/6/7/   43# 4/1/7/3/   44# 4/1/7/6/
    3  6 15 10     4  5 16  9     3  6 15 10     4  5 16  9
   16  9  4  5    15 10  3  6    13 12  1  8    14 11  2  7
    2  7 14 11     1  8 13 12     2  7 14 11     1  8 13 12
   13 12  1  8    14 11  2  7    16  9  4  5    15 10  3  6

  45# 4/3/1/2/   46# 4/3/1/7/   47# 4/6/1/2/   48# 4/6/1/7/
    1  8 11 14     2  7 12 13     5  4 15 10     6  3 16  9
   12 13  2  7    11 14  1  8    16  9  6  3    15 10  5  4
    6  3 16  9     5  4 15 10     2  7 12 13     1  8 11 14
   15 10  5  4    16  9  6  3    11 14  1  8    12 13  2  7

 [Count of Compositions = 48]

 ** DATA Sentences in Characters **
/**/
short tlu[9][16]={
 {0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0},
 {0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0},
 {0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1},
```

```
 {0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1},
 {1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0},
 {1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0},
 {1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1},
 {1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1},
 {-1}};
/**/
short tans[49][4]={
 {1,2,4,3}, {1,2,4,6}, {1,4,2,3}, {1,4,2,6},
 {1,4,3,2}, {1,4,3,7}, {1,4,6,2}, {1,4,6,7},
 {1,4,7,3}, {1,4,7,6}, {1,7,4,3}, {1,7,4,6},
 {2,1,4,3}, {2,1,4,6}, {2,4,1,3}, {2,4,1,6},
 {2,4,3,1}, {2,4,3,8}, {2,4,6,1}, {2,4,6,8},
 {2,4,8,3}, {2,4,8,6}, {2,8,4,3}, {2,8,4,6},
 {4,1,2,3}, {4,1,2,6}, {4,1,3,2}, {4,1,3,7},
 {4,1,6,2}, {4,1,6,7}, {4,1,7,3}, {4,1,7,6},
 {4,2,1,3}, {4,2,1,6}, {4,2,3,1}, {4,2,3,8},
 {4,2,6,1}, {4,2,6,8}, {4,2,8,3}, {4,2,8,6},
 {4,3,1,2}, {4,3,1,7}, {4,3,2,1}, {4,3,2,8},
 {4,3,7,1}, {4,3,7,8}, {4,3,8,2}, {4,3,8,7},
 {-1}};
/**/
```

Let's make a little experiment in order to know what to do and how to do with it in the case of order 4.

As you see above, you can save those "DATA Sentences" in place of the whole solution list. You can express each solution by the list of 4 numbers of Used Units. How smart it is!

DATA sentences of number list are written in character codes with each comma ',' for termination. This form is widely accepted by various typical application softwares.

Why don't you write your DATA sentences like these? You don't have to save that amount of long list any longer. You can easily re-use your solution data for another job, can't you?

How can you read and re-use those DATA sentences? Let me show you a little example of my program list.

```
/*** Read the DATA Strings, Decode them and Re-compose the  ***/
/** Solutions of 'Composite and Complete' Magic Squares' 4x4 **/
/*** Made by New Euler's Method of Binary System: K. Setsuda **/
/**/
/* 'CES4RdPr.c' */
/**/
#include <stdio.h>
/**/
short int cnt, cnt2, cnt3;
short cntlu, cntans;
short uum[9];
short anm[4][22];
/**/
short tlu[9][16]={
 {0,0,1,1,1,1,0,0,0,0,1,1,1,1,0,0},
 {0,1,0,1,0,1,0,1,1,0,1,0,1,0,1,0},
 {0,1,0,1,1,0,1,0,1,0,1,0,0,1,0,1},
 {0,1,1,0,1,0,0,1,0,1,1,0,1,0,0,1},
 {1,0,0,1,0,1,1,0,1,0,0,1,0,1,1,0},
```

```
 {1,0,1,0,0,1,0,1,0,1,0,1,1,0,1,0},
 {1,0,1,0,1,0,1,0,0,1,0,1,0,1,0,1},
 {1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1},
 {-1}};
/**/
short tans[49][4]={
 {1,2,4,3}, {1,2,4,6}, {1,4,2,3}, {1,4,2,6},
 {1,4,3,2}, {1,4,3,7}, {1,4,6,2}, {1,4,6,7},
 {1,4,7,3}, {1,4,7,6}, {1,7,4,3}, {1,7,4,6},
 {2,1,4,3}, {2,1,4,6}, {2,4,1,3}, {2,4,1,6},
 {2,4,3,1}, {2,4,3,8}, {2,4,6,1}, {2,4,6,8},
 {2,4,8,3}, {2,4,8,6}, {2,8,4,3}, {2,8,4,6},
 {4,1,2,3}, {4,1,2,6}, {4,1,3,2}, {4,1,3,7},
 {4,1,6,2}, {4,1,6,7}, {4,1,7,3}, {4,1,7,6},
 {4,2,1,3}, {4,2,1,6}, {4,2,3,1}, {4,2,3,8},
 {4,2,6,1}, {4,2,6,8}, {4,2,8,3}, {4,2,8,6},
 {4,3,1,2}, {4,3,1,7}, {4,3,2,1}, {4,3,2,8},
 {4,3,7,1}, {4,3,7,8}, {4,3,8,2}, {4,3,8,7},
 {-1}};
/**/
void rddcod(void), prlunit(void), prans(void);
void pr4ans(void), pr3ans(void), pr2ans(void), pr1ans(void);
void uumon(void), prdata(void);
/**/
/* Main Program: Read, Re-compose and Print */
int main(){
  printf("\n*** Read the DATA Strings, Decode them and Re-compose the  ***\n");
  printf("** Solutions of 'Composite and Complete' Magic Squares' 4x4 **\n");
  printf("*** Made by New Euler's Method of Binary System: K. Setsuda **\n");
  rddcod();
  printf("\n [Latin Units]\n\n");
  prlunit();
  printf("\n [Count of Latin Units = %d]\n",cntlu);
  cnt=0; cnt2=0; cnt3=0;
  printf("\n [Re-compositions: List Number# Used Units////]\n\n");
  prans();
  if(cnt3==3){pr3ans();}
  if(cnt3==2){pr2ans();}
  if(cnt3==1){pr1ans();}
  printf(" [Count of Compositions = %d/%d]\n",cnt2,cnt);
  printf("\n [Used Unit Monitor]\n");
  uumon();
  printf("\n [DATA Strings]\n");
  prdata();
  printf("\n OK!\n");
 return 0;
}
/**/
/* Read the Data String and Decode them */
void rddcod(){
 short t,n,v;
  t=0; while(tlu[t][0]>=0){t++;}
  cntlu=t;
/**/
  t=0; while(tans[t][0]>=0){t++;}
  cntans=t;
  for(t=0;t<cntans;t++){
    for(n=0;n<4;n++){
      v=tans[t][n]; tans[t][n]=v-1;
```

```
  }}
}
/**/
/* Print the Latin Units */
void prlunit(){
 short t,l,l4,m,n;
  for(t=0;t<cntlu;t=t+4){
    printf("%9d/%9d/%9d/%9d/\n",t+1,t+2,t+3,t+4);
    for(l=0;l<4;l++){l4=l*4;
      for(m=t;m<(t+4);m++){
        printf("  ");
        for(n=0;n<4;n++){printf("%2d",tlu[m][l4+n]);}
      }
      printf("\n");
    }
  }
}
/**/
/* Print The Answer */
void prans(){
 short t,n;
 short tu[4], nm[22];
  for(n=0;n<cntlu;n++){uum[n]=0;}
  for(t=0;t<cntans;t++){cnt++;
    for(n=0;n<4;n++){tu[n]=tans[t][n];}
    for(n=0;n<16;n++){
      nm[n+1]=tlu[tu[0]][n]*8+tlu[tu[1]][n]*4+tlu[tu[2]][n]*2+tlu[tu[3]][n]+1;
    }
  /*if((nm[1]==1)&&(nm[4]<nm[13])&&(nm[2]<nm[4])&&(nm[5]<nm[13])){*/
    if((nm[1]==1)&&(nm[11]==16)){
      cnt2++;
      anm[cnt3][0]=cnt;
      for(n=1;n<17;n++){anm[cnt3][n]=nm[n];}
      for(n=0;n<4;n++){
        anm[cnt3][17+n]=tu[n]+1;
        uum[tu[n]]++;}
      anm[cnt3][21]=cnt2;
      cnt3++;
      if(cnt3==2){pr2ans(); cnt3=0;}
    }
  }
}
/**/
/* Print 4 Answers */
void pr4ans(){
 short s,l,l4,n;
  for(s=0;s<4;s++){
    printf("%5d#%2d/%d/%d/%d/",
      anm[s][21],anm[s][17],anm[s][18],anm[s][19],anm[s][20]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(s=0;s<4;s++){
        printf("   ");
        for(n=1;n<5;n++){printf("%3d",anm[s][l4+n]);}
    }
    printf("\n");
  }
  printf("\n");
}
```

```
/**/
/* Print 3 Answers */
void pr3ans(){
 short s,l,l4,n;
  for(s=0;s<3;s++){
    printf("%5d#%2d/%d/%d/%d/",
      anm[s][21],anm[s][17],anm[s][18],anm[s][19],anm[s][20]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(s=0;s<3;s++){
        printf("   ");
        for(n=1;n<5;n++){printf("%3d",anm[s][l4+n]);}
    }
    printf("\n");
  }
  printf("\n");
}
/**/
/* Print 2 Answers */
void pr2ans(){
 short s,l,l4,m,n;
 short tu[4];
  for(s=0;s<2;s++){
    printf("%5d/%4d/%4d/%4d/%12d# ",
      anm[s][17],anm[s][18],anm[s][19],anm[s][20],anm[s][21]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(s=0;s<2;s++){
      for(n=0;n<4;n++){tu[n]=anm[s][n+17]-1;}
      printf(" ");
      for(m=0;m<4;m++){
        printf(" ");
        for(n=0;n<4;n++){printf("%d",tlu[tu[m]][l4+n]);}
      }
      printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[s][l4+n]);}
      printf(" ");
    }
    printf("\n");
  }
  printf("\n");
}
/**/
/* Print The Answer */
void pr1ans(){
 short l,l4,m,n;
 short tu[4];
  printf("%9d/%8d/%8d/%8d/%12d#\n",
    anm[0][17],anm[0][18],anm[0][19],anm[0][20],anm[0][21]);
  for(l=0;l<4;l++){l4=l*4;
    for(n=0;n<4;n++){tu[n]=anm[0][n+17]-1;}
    printf(" ");
    for(m=0;m<4;m++){
      printf(" ");
      for(n=0;n<4;n++){printf("%2d",tlu[tu[m]][l4+n]);}
    }
    printf(" ");
    for(n=1;n<5;n++){printf("%3d",anm[0][l4+n]);}
    printf("\n");
```

```
  }
  printf("\n");
}
/**/
/* Used Unit Monitor */
void uumon(){
 short m,n;
  for(n=0;n<cntlu;n++){m=uum[n];
    if(m>0){printf("%2d/%d,  ",n+1,m);}
  }
  printf("\n");
}
/**/
/* Print Data Strings */
void prdata(){
 short t,n,v;
  printf("/**/\n");
  printf("short tlu[%d][16]={\n",cntlu+1);
  for(t=0;t<cntlu;t++){
    printf(" {");
    for(n=0;n<16;n++){
      printf("%d",tlu[t][n]);
      if(n<15){printf(",");}
    }
    printf("},\n");
  }
  printf(" {-1}};\n");
  printf("/**/\n");
/**/
  printf("short tans[%d][4]={\n",cntans+1);
  for(t=0;t<cntans;t++){
    printf(" {");
    for(n=0;n<4;n++){
      v=tans[t][n]+1;
      printf("%d",v);
      if(n<3){printf(",");}
    }
    printf("},");
    if((t+1)%4==0){printf("\n");}
  }
  printf(" {-1}};\n");
  printf("/**/\n");
}
/**/
```

This program is just written for printing the whole list of solutions recomposed. I put the print program of DATA sentences in the end of the list above.

I wish you can easily understand all, but pay your special attention to the difference of number systems between the inside calculation and its outside expression. Integer series beginning with 0 are used inside, but the series of Natural Numbers beginning with 1 are used for outside.

**(2) For Data Compression**: If you want to make your DATA sentences more compact, especially of the list of Latin Units, how about reducing the data size by Binary Number System?

Take your look at the DATA Sentences again. Every list of Latin Units contains only '0' and '1'. It is natural because everything is made of binary numbers here. Why don't

you take 4 numbers in each row and assume them as a whole number of 4 digits and express it in decimal? Take {0, 1, 1, 0}, for instance, and calculate as follows:

```
((0*2+1)*2+1)*2+0 = 6(Decimal)
```

Then you can express each layer by 4 decimal numbers like {6, 9, 8, 9}. How smart it is!

If you want to make it the most compact, take all 16 elements of each layer, and calculate the single value of those in decimal. You can take {0,1,1,0, 1,0,0,1, 0,1,1,0, 1,0,0,1}, for instance, and calculate as follows:

```
((((((((((((((0*2+1)*2+1)*2+0)*2+1)*2+0)*2+0)*2+1)*2+0)*2+1)*2+1)*2+0)
  *2+1)*2+0)*2+0)*2+1 = 26985(Decimal)
```

You can thus express a whole layer by a single number in decimal, and only have the 8 numbers for the whole list of Latin Units. How great!

I would like to skip my explaining programs for 'compression' and 'expansion'. I just want you to think about it by yourself.

The next list shows the part of my result of composition: "3-type Simultaneous" Euler Squares of order 8: 'Composite', Self-complementary and Pan-diagonal.

If you want to save 360 "Standard Solutions" with n1=1 to outer memory devices, how can you do that? How can you write your DATA sentences for another program?

```
** 3-type Simultaneous Magic Squares 8x8: 'Composite',
 Self-Complementary and Pan-diagonal by New Euler's Method **
[Latin Units]
      1/       2/       3/       4/       5/       6/
 01010101 01010101 01011010 01100110 01010101 00111100
 10101010 10101010 10100101 10011001 01010101 11000011
 01010101 10101010 01011010 01100110 10101010 00111100
 10101010 01010101 10100101 10011001 10101010 11000011
 10101010 01010101 01011010 01100110 10101010 00111100
 01010101 10101010 10100101 10011001 10101010 11000011
 10101010 10101010 01011010 01100110 01010101 00111100
 01010101 01010101 10100101 10011001 01010101 11000011
      7/       8/       9/      10/      11/      12/
 11000011 10101010 10011001 10100101 10101010 10101010
 00111100 10101010 01100110 01011010 01010101 01010101
 11000011 01010101 10011001 10100101 01010101 10101010
 00111100 01010101 01100110 01011010 10101010 01010101
 11000011 01010101 10011001 10100101 10101010 01010101
 00111100 01010101 01100110 01011010 01010101 10101010
 11000011 10101010 10011001 10100101 01010101 01010101
 00111100 10101010 01100110 01011010 10101010 10101010
[Coumt of Latin Units = 12]

[Reference Table for the Best Combinations]
 *|  1  2  3  4  5  6  7  8  9 10 11 12
--+-----------------------------------
 1| 64 32 32 32 32 32 32 32 32 32 32  0
 2| 32 64 32 32 32 32 32 32 32 32  0 32
 3| 32 32 64 32 32 32 32 32 32  0 32 32
 4| 32 32 32 64 32 32 32 32  0 32 32 32
 5| 32 32 32 32 64 32 32  0 32 32 32 32
 6| 32 32 32 32 32 64  0 32 32 32 32 32
```

```
 7| 32 32 32 32 32  0 64 32 32 32 32 32
 8| 32 32 32 32  0 32 32 64 32 32 32 32
 9| 32 32 32  0 32 32 32 32 64 32 32 32
10| 32 32  0 32 32 32 32 32 32 64 32 32
11| 32  0 32 32 32 32 32 32 32 32 64 32
12|  0 32 32 32 32 32 32 32 32 32 32 64
```

** **List of Compositions**: Used Units//  List Number# **

```
     1/       2/       3/       4/       5/       6/                      1#
01010101 01010101 01011010 01100110 01010101 00111100   1 63  6 60 10 56 13 51
10101010 10101010 10100101 10011001 01010101 11000011  62  4 57  7 53 11 50 16
01010101 10101010 01011010 01100110 10101010 00111100  19 45 24 42 28 38 31 33
10101010 01010101 10100101 10011001 10101010 11000011  48 18 43 21 39 25 36 30
10101010 01010101 01011010 01100110 10101010 00111100  35 29 40 26 44 22 47 17
01010101 10101010 10100101 10011001 10101010 11000011  32 34 27 37 23 41 20 46
10101010 10101010 01011010 01100110 01010101 00111100  49 15 54 12 58  8 61  3
01010101 01010101 10100101 10011001 01010101 11000011  14 52  9 55  5 59  2 64

     1/       2/       3/       5/       4/       6/                      2#
01010101 01010101 01011010 01010101 01100110 00111100   1 63  4 62 10 56 11 53
10101010 10101010 10100101 01010101 10011001 11000011  60  6 57  7 51 13 50 16
01010101 10101010 01011010 10101010 01100110 00111100  21 43 24 42 30 36 31 33
10101010 01010101 10100101 10101010 10011001 11000011  48 18 45 19 39 25 38 28
10101010 01010101 01011010 10101010 01100110 00111100  37 27 40 26 46 20 47 17
01010101 10101010 10100101 10101010 10011001 11000011  32 34 29 35 23 41 22 44
10101010 10101010 01011010 01010101 01100110 00111100  49 15 52 14 58  8 59  5
01010101 01010101 10100101 01010101 10011001 11000011  12 54  9 55  3 61  2 64

     1/       2/       3/       5/       6/       4/                      3#
01010101 01010101 01011010 01010101 00111100 01100110   1 62  4 63 11 56 10 53
10101010 10101010 10100101 01010101 11000011 10011001  60  7 57  6 50 13 51 16
01010101 10101010 01011010 10101010 00111100 01100110  21 42 24 43 31 36 30 33
10101010 01010101 10100101 10101010 11000011 10011001  48 19 45 18 38 25 39 28
10101010 01010101 01011010 10101010 00111100 01100110  37 26 40 27 47 20 46 17
01010101 10101010 10100101 10101010 11000011 10011001  32 35 29 34 22 41 23 44
10101010 10101010 01011010 01010101 00111100 01100110  49 14 52 15 59  8 58  5
01010101 01010101 10100101 01010101 11000011 10011001  12 55  9 54  2 61  3 64

 No.4#  1/ 2/ 4/ 3/ 5/ 6/  No.5#  1/ 2/ 4/ 5/ 3/ 6/  No.6#  1/ 2/ 4/ 5/ 6/ 3/
   1 63 10 56  6 60 13 51    1 63 10 56  4 62 11 53    1 62 11 56  4 63 10 53
  62  4 53 11 57  7 50 16   60  6 51 13 57  7 50 16   60  7 50 13 57  6 51 16
  19 45 28 38 24 42 31 33   21 43 30 36 24 42 31 33   21 42 31 36 24 43 30 33
  48 18 39 25 43 21 36 30   48 18 39 25 45 19 38 28   48 19 38 25 45 18 39 28
  35 29 44 22 40 26 47 17   37 27 46 20 40 26 47 17   37 26 47 20 40 27 46 17
  32 34 23 41 27 37 20 46   32 34 23 41 29 35 22 44   32 35 22 41 29 34 23 44
  49 15 58  8 54 12 61  3   49 15 58  8 52 14 59  5   49 14 59  8 52 15 58  5
  14 52  5 59  9 55  2 64   12 54  3 61  9 55  2 64   12 55  2 61  9 54  3 64

 No.7#  1/ 2/ 5/ 3/ 4/ 6/  No.8#  1/ 2/ 5/ 3/ 6/ 4/  No.9#  1/ 2/ 5/ 4/ 3/ 6/
   1 63  4 62  6 60  7 57    1 62  4 63  7 60  6 57    1 63  6 60  4 62  7 57
  56 10 53 11 51 13 50 16   56 11 53 10 50 13 51 16   56 10 51 13 53 11 50 16
  25 39 28 38 30 36 31 33   25 38 28 39 31 36 30 33   25 39 30 36 28 38 31 33
  48 18 45 19 43 21 42 24   48 19 45 18 42 21 43 24   48 18 43 21 45 19 42 24
  41 23 44 22 46 20 47 17   41 22 44 23 47 20 46 17   41 23 46 20 44 22 47 17
  32 34 29 35 27 37 26 40   32 35 29 34 26 37 27 40   32 34 27 37 29 35 26 40
  49 15 52 14 54 12 55  9   49 14 52 15 55 12 54  9   49 15 54 12 52 14 55  9
   8 58  5 59  3 61  2 64    8 59  5 58  2 61  3 64    8 58  3 61  5 59  2 64
```

```
  10#  1/ 2/ 5/ 4/ 6/ 3/    11#  1/ 2/ 5/ 6/ 3/ 4/    12#  1/ 2/ 5/ 6/ 4/ 3/
   1 62  7 60  4 63  6 57     1 60  6 63  7 62  4 57     1 60  7 62  6 63  4 57
  56 11 50 13 53 10 51 16    56 13 51 10 50 11 53 16    56 13 50 11 51 10 53 16
  25 38 31 36 28 39 30 33    25 36 30 39 31 38 28 33    25 36 31 38 30 39 28 33
  48 19 42 21 45 18 43 24    48 21 43 18 42 19 45 24    48 21 42 19 43 18 45 24
  41 22 47 20 44 23 46 17    41 20 46 23 47 22 44 17    41 20 47 22 46 23 44 17
  32 35 26 37 29 34 27 40    32 37 27 34 26 35 29 40    32 37 26 35 27 34 29 40
  49 14 55 12 52 15 54  9    49 12 54 15 55 14 52  9    49 12 55 14 54 15 52  9
   8 59  2 61  5 58  3 64     8 61  3 58  2 59  5 64     8 61  2 59  3 58  5 64

  . . . . . .

  355#  5/ 6/ 4/ 1/ 2/ 3/   356#  5/ 6/ 4/ 1/ 3/ 2/   357#  5/ 6/ 4/ 2/ 1/ 3/
   1 48 25 56 18 63 10 39     1 48 25 56 19 62 11 38     1 48 25 56 18 63 10 39
  32 49  8 41 15 34 23 58    32 49  8 41 14 35 22 59    32 49  8 41 15 34 23 58
  35 14 59 22 52 29 44  5    34 15 58 23 52 29 44  5    37 12 61 20 54 27 46  3
  62 19 38 11 45  4 53 28    63 18 39 10 45  4 53 28    60 21 36 13 43  6 51 30
  37 12 61 20 54 27 46  3    37 12 61 20 55 26 47  2    35 14 59 22 52 29 44  5
  60 21 36 13 43  6 51 30    60 21 36 13 42  7 50 31    62 19 38 11 45  4 53 28
   7 42 31 50 24 57 16 33     6 43 30 51 24 57 16 33     7 42 31 50 24 57 16 33
  26 55  2 47  9 40 17 64    27 54  3 46  9 40 17 64    26 55  2 47  9 40 17 64

  358#  5/ 6/ 4/ 2/ 3/ 1/   359#  5/ 6/ 4/ 3/ 1/ 2/   360#  5/ 6/ 4/ 3/ 2/ 1/
   1 48 25 56 19 62 11 38     1 48 25 56 21 60 13 36     1 48 25 56 21 60 13 36
  32 49  8 41 14 35 22 59    32 49  8 41 12 37 20 61    32 49  8 41 12 37 20 61
  37 12 61 20 55 26 47  2    34 15 58 23 54 27 46  3    35 14 59 22 55 26 47  2
  60 21 36 13 42  7 50 31    63 18 39 10 43  6 51 30    62 19 38 11 42  7 50 31
  34 15 58 23 52 29 44  5    35 14 59 22 55 26 47  2    34 15 58 23 54 27 46  3
  63 18 39 10 45  4 53 28    62 19 38 11 42  7 50 31    63 18 39 10 43  6 51 30
   6 43 30 51 24 57 16 33     4 45 28 53 24 57 16 33     4 45 28 53 24 57 16 33
  27 54  3 46  9 40 17 64    29 52  5 44  9 40 17 64    29 52  5 44  9 40 17 64
 [Count = 360/5760]
```

If you want to have 'Standard Solutions' with n1=1, you may only have the first 6 Latin Units in all. Therefore you can write your DATA sentences in half size.

Take 8 numbers in each row of each layer, assume them as a whole number of 8 digits and calculate a single value for it in decimal. And you can express each layer with 8 numbers in decimal.

Every number list of Used Units for each solution could be made compact into a single decimal number, since it has no numbers more than 6 in the list, you can compress each data by PNS of Base 6.

You can write your result of calculation in such DATA sentences just as in the list in the next page below.

How can you read those DATA sentences in another program?

The next list shows a sample program for that kind of job.

```
/** Read DATA Strings, Decode and Recompose them back into ***/
/** Three-Type Simultaneous Magic Squares' 8x8: Composite, ***/
/**  Self-Complementary and Pan-diagonal: by Kanji Setsuda  **/
/** 'CES8Sml3D.c' built first on Oct.30, 2003 with MWCW **/
/** Revised on Mar.25, 2007 with MacOSX and Xcode 2.2.1 **/
/**/
#include <stdio.h>
/**/
short int cnt, cnt2, cnt3;
short cntlu, cntans;
short anm[4][72];
```

```
short uum[16];
short tlu[7][64], ntlu[7][64];
short tans[361][6], ntans[361][6];
/**/
/* Data Sentences */
short lutbl[7][8]={
 { 85,170, 85,170,170, 85,170, 85},
 { 85,170,170, 85, 85,170,170, 85},
 { 90,165, 90,165, 90,165, 90,165},
 {102,153,102,153,102,153,102,153},
 { 85, 85,170,170,170,170, 85, 85},
 { 60,195, 60,195, 60,195, 60,195},
 {-208}};
/**/
long anstbl[361]={
  1865,  1895,  1905,  2045,  2105,  2120,  2255,  2265,  2285,  2300,
  2355,  2360,  2945,  2975,  2985,  3305,  3395,  3415,  3515,  3525,
  3575,  3595,  3645,  3655,  4205,  4265,  4280,  4385,  4475,  4495,
  4805,  4820,  4835,  4855,  4940,  4945,  5495,  5505,  5525,  5540,
  5595,  5600,  5675,  5685,  5735,  5755,  5805,  5815,  5885,  5900,
  5915,  5935,  6020,  6025,  6315,  6320,  6345,  6355,  6380,  6385,
  8345,  8375,  8385,  8525,  8585,  8600,  8735,  8745,  8765,  8780,
  8835,  8840, 10505, 10535, 10545, 11045, 11165, 11190, 11255, 11265,
 11345, 11370, 11415, 11430, 11765, 11825, 11840, 12125, 12245, 12270,
 12545, 12560, 12605, 12630, 12710, 12720, 13055, 13065, 13085, 13100,
 13155, 13160, 13415, 13425, 13505, 13530, 13575, 13590, 13625, 13640,
 13685, 13710, 13790, 13800, 14055, 14060, 14115, 14130, 14150, 14160,
 15905, 15935, 15945, 16265, 16355, 16375, 16475, 16485, 16535, 16555,
 16605, 16615, 16985, 17015, 17025, 17525, 17645, 17670, 17735, 17745,
 17825, 17850, 17895, 17910, 19505, 19595, 19615, 19685, 19805, 19830,
 20315, 20335, 20345, 20370, 20485, 20490, 20795, 20805, 20855, 20875,
 20925, 20935, 20975, 20985, 21065, 21090, 21135, 21150, 21395, 21415,
 21425, 21450, 21565, 21570, 21825, 21835, 21855, 21870, 21925, 21930,
 23645, 23705, 23720, 23825, 23915, 23935, 24245, 24260, 24275, 24295,
 24380, 24385, 24725, 24785, 24800, 25085, 25205, 25230, 25505, 25520,
 25565, 25590, 25670, 25680, 25985, 26075, 26095, 26165, 26285, 26310,
 26795, 26815, 26825, 26850, 26965, 26970, 28565, 28580, 28595, 28615,
 28700, 28705, 28745, 28760, 28805, 28830, 28910, 28920, 28955, 28975,
 28985, 29010, 29125, 29130, 29600, 29605, 29630, 29640, 29665, 29670,
 31415, 31425, 31445, 31460, 31515, 31520, 31595, 31605, 31655, 31675,
 31725, 31735, 31805, 31820, 31835, 31855, 31940, 31945, 32235, 32240,
 32265, 32275, 32300, 32305, 32495, 32505, 32525, 32540, 32595, 32600,
 32855, 32865, 32945, 32970, 33015, 33030, 33065, 33080, 33125, 33150,
 33230, 33240, 33495, 33500, 33555, 33570, 33590, 33600, 33755, 33765,
 33815, 33835, 33885, 33895, 33935, 33945, 34025, 34050, 34095, 34110,
 34355, 34375, 34385, 34410, 34525, 34530, 34785, 34795, 34815, 34830,
 34885, 34890, 35045, 35060, 35075, 35095, 35180, 35185, 35225, 35240,
 35285, 35310, 35390, 35400, 35435, 35455, 35465, 35490, 35605, 35610,
 36080, 36085, 36110, 36120, 36145, 36150, 37635, 37640, 37665, 37675,
 37700, 37705, 37815, 37820, 37875, 37890, 37910, 37920, 38025, 38035,
 38055, 38070, 38125, 38130, 38240, 38245, 38270, 38280, 38305, 38310,
 -606};
/**/
/* Sub-Routines */
void rddcod(void), prlunit(void), prans(void);
void pr3ans(void);
void prdata(void), uumon(void);
/**/
/* Main Program: Reset All and Go! */
```

```
int main(){
  printf("\n** Read DATA Strings, Decode and Recompose them back into **\n");
  printf("** Three-Type Simultaneous Magic Squares' 8x8: Composite, **\n");
  printf("**  Self-Complementary and Pan-diagonal: by Kanji Setsuda **\n");
  cnt=0;
  rddcod();
  printf("[Latin Units]\n");
  prlunit();
  cnt=0; cnt2=0; cnt3=0;
  printf("\n [Re-compositions: List Number# used units//////]\n");
  prans();
  printf(" [Count = %d/%d#]\n",cnt2,cnt);
  cnt=0;
  printf("\n [Used Unit Monitor]\n");
  uumon();
  printf("\n [DATA Strings]\n");
  prdata();
  printf("\n OK!\n");
 return 0;
}
/**/
/* Read and Decode the Data Strings */
void rddcod(){
 long int v;
 short t,m,n,npos,bas,pos;
 short dnm[8];
  t=0; while(lutbl[t][0]>=0){t++;};
  cntlu=t; n=lutbl[cntlu][0]*(-1);
  bas=n/100; pos=n%100;
  for(t=0;t<cntlu;t++){
    for(n=0;n<pos;n++){npos=n*pos;
      v=lutbl[t][n]; m=pos-1;
      while(m>=0){
        dnm[m]=v%bas; v=v/bas;
      m--;}
      for(m=0;m<pos;m++){tlu[t][npos+m]=dnm[m];}
    }
  }
  tlu[cntlu][0]=-1;
/**/
  t=0; while(anstbl[t]>=0){t++;};
  cntans=t; v=anstbl[cntans]*(-1);
  bas=v/100; pos=v%100;
  for(t=0;t<cntans;t++){
    v=anstbl[t]; m=pos-1;
    while(m>=0){
      dnm[m]=v%bas; v=v/bas;
    m--;}
    for(n=0;n<pos;n++){tans[t][n]=dnm[n];}
  }
  tans[cntans][0]=-1;
}
/**/
/* Print the Latin Units */
void prlunit(){
 short t,u,m,n,l,l8;
  if(cntlu>6){u=6;}else{u=cntlu;}
  for(t=0;t<cntlu;t=t+u){
    for(n=1;n<=u;n++){printf("%9d/",t+n);}
    printf("\n");
    for(l=0;l<8;l++){l8=l*8;
      for(m=t;m<(t+u);m++){
        printf("  ");
```

```
        for(n=0;n<8;n++){printf("%d",tlu[m][l8+n]);}
      }
      printf("\n");
    }
  }
  printf("[Coumt of Latin Units = %d]\n",cntlu);
}
/**/
/* Recompose and Print the Answers */
void prans(){
 short t,m,n;
 short tu1,tu2,tu3,tu4,tu5,tu6;
  for(t=0;t<cntans;t++){cnt++;
    tu1=tans[t][0]; tu2=tans[t][1]; tu3=tans[t][2];
    tu4=tans[t][3]; tu5=tans[t][4]; tu6=tans[t][5];
    for(n=0;n<64;n++){
      m=tlu[tu1][n]*32+tlu[tu2][n]*16+tlu[tu3][n]*8+tlu[tu4][n]*4+tlu[tu5][n]*2+tlu[tu6][n]+1;
      anm[cnt3][n+1]=m;}
    anm[cnt3][65]=tu1+1; anm[cnt3][66]=tu2+1; anm[cnt3][67]=tu3+1;
    anm[cnt3][68]=tu4+1; anm[cnt3][69]=tu5+1; anm[cnt3][70]=tu6+1;
    if((anm[cnt3][1]==1)&&(anm[cnt3][64]==64)){
      uum[tu1]++; uum[tu2]++; uum[tu3]++; uum[tu4]++; uum[tu5]++; uum[tu6]++;
      for(n=0;n<6;n++){ntans[cnt2][n]=tans[t][n];}
      cnt2++; anm[cnt3][0]=cnt2;
      cnt3++;
      if(cnt3==3){pr3ans(); cnt3=0;}
    }
  }
  cntans=cnt2;
}
/**/
/* Print 3 Answers */
void pr3ans(){
  short l,l8,m,n;
    for(m=0;m<3;m++){
      printf("%4d#%4d/%2d/%2d/%2d/%2d/%2d/ ",
        anm[m][0],anm[m][65],anm[m][66],anm[m][67],anm[m][68],anm[m][69],anm[m][70]);}
    printf("\n");
    for(l=0;l<8;l++){l8=l*8;
      for(m=0;m<3;m++){
        printf("  ");
        for(n=1;n<9;n++){printf("%3d",anm[m][l8+n]);}}
      printf("\n");
    }
    printf("\n");
}
/**/
/* Used Unit Monitor */
void uumon(){
 short m,n,p;
  for(n=0;n<cntlu;n++){
    m=uum[n]; uum[n]=-1;
    if(m>0){
      printf("%2d/%d, ",n+1,m);
      for(p=0;p<64;p++){ntlu[cnt][p]=tlu[n][p];}
      uum[n]=cnt;
      cnt++;
    }
  }
  printf("\n");
  cntlu=cnt;
/**/
  for(n=0;n<cntans;n++){
    for(m=0;m<6;m++){
```

```
      p=ntans[n][m];
      ntans[n][m]=uum[p];
  }}
}
/**/
/* Print the Data Strings */
void prdata(){
 long int v;
 short t,m,n,pos,bas;
 short tu1,tu2,tu3,tu4,tu5,tu6;
  pos=8; bas=2;
  printf("/**/\n");
  printf("short lutbl[%d][8]={\n",cntlu+1);
  for(t=0;t<cntlu;t++){
    printf(" {");
    for(m=0;m<64;m=m+8){v=0;
      for(n=0;n<(pos-1);n++){v=(v+ntlu[t][m+n])*bas;}
      v=v+ntlu[t][m+pos-1];
      printf("%3d",v);
      if(m<56){printf(",",v);}
    }
    printf("},\n");
  }
  printf(" {%d}};\n",(-1)*bas*100-pos);
/**/
  pos=6; bas=cntlu;
  printf("/**/\n");
  printf("long anstbl[%d]={\n",cntans+1);
  for(t=0;t<cntans;t++){
    tu1=ntans[t][0]; tu2=ntans[t][1]; tu3=ntans[t][2];
    tu4=ntans[t][3]; tu5=ntans[t][4]; tu6=ntans[t][5];
    v=((((tu1*bas+tu2)*bas+tu3)*bas+tu4)*bas+tu5)*bas+tu6;
    printf("%5d,",v);
    if((t+1)%10==0){printf("\n");}
  }
  printf("%d};\n",(-1)*bas*100-pos);
  printf("/**/\n");
}
/**/
```

You could compress all the solution data with 8x8x360 numbers into the DATA sentences in one page size. That will help you save your memories, papers, time and mental energy, won't it? What a relief!

You can easily re-use the whole data for another program. I hope it will help you study Magic Squares further, either analytically or synthetically.

Please change the part of '`prans();`' procedure above, and make your own program and write it in place of my inequality conditions.

It seems these techniques of New Euler's Method may be applicable and effective in the field of "Cryptograph", but I am not an expert of it by any means. I would like to skip that topic here.

**(3) For Analytical Study**: You can apply our New Euler's Method for another study of Euler Squares, for instance, analytical study directly on its Latin Units and the number list of Used Units themselves.

Let me show you how it could be done in the case of order 4.

Take your look at the next new list of 48 solutions of Pan-diagonal Euler Squares 4x4.

**** Pan-diagonal MS44 Made by New Euler's Method ****

[Latin Units]

```
      1/          2/          3/          4/
 0 0 1 1     0 1 0 1     0 1 0 1     0 1 1 0
 1 1 0 0     0 1 0 1     1 0 1 0     1 0 0 1
 0 0 1 1     1 0 1 0     1 0 1 0     0 1 1 0
 1 1 0 0     1 0 1 0     0 1 0 1     1 0 0 1
      5/          6/          7/          8/
 1 0 0 1     1 0 1 0     1 0 1 0     1 1 0 0
 0 1 1 0     0 1 0 1     1 0 1 0     0 0 1 1
 1 0 0 1     0 1 0 1     0 1 0 1     1 1 0 0
 0 1 1 0     1 0 1 0     0 1 0 1     0 0 1 1
```

[**List of Compositions**: List Number# Used Units////]

```
  1# 1/2/4/3/     2# 1/4/2/3/     3# 1/4/3/2/     4# 2/1/4/3/
   1  8 11 14      1  8 13 12      1  8 13 12      1 12  7 14
  12 13  2  7     14 11  2  7     15 10  3  6      8 13  2 11
   6  3 16  9      4  5 16  9      4  5 16  9     10  3 16  5
  15 10  5  4     15 10  3  6     14 11  2  7     15  6  9  4
  5# 2/4/1/3/     6# 2/4/3/1/     7# 4/1/2/3/     8# 4/1/3/2/
   1 14  7 12      1 15  6 12      1 12 13  8      1 12 13  8
   8 11  2 13      8 10  3 13     14  7  2 11     15  6  3 10
  10  5 16  3     11  5 16  2      4  9 16  5      4  9 16  5
  15  4  9  6     14  4  9  7     15  6  3 10     14  7  2 11
  9# 4/2/1/3/    10# 4/2/3/1/    11# 4/3/1/2/    12# 4/3/2/1/
   1 14 11  8      1 15 10  8      1 14 11  8      1 15 10  8
  12  7  2 13     12  6  3 13     15  4  5 10     14  4  5 11
   6  9 16  3      7  9 16  2      6  9 16  3      7  9 16  2
  15  4  5 10     14  4  5 11     12  7  2 13     12  6  3 13
 13# 1/2/4/6/    14# 1/4/2/6/    15# 1/4/3/7/    16# 2/1/4/6/
   2  7 12 13      2  7 14 11      2  7 14 11      2 11  8 13
  11 14  1  8     13 12  1  8     16  9  4  5      7 14  1 12
   5  4 15 10      3  6 15 10      3  6 15 10      9  4 15  6
  16  9  6  3     16  9  4  5     13 12  1  8     16  5 10  3
 17# 2/4/1/6/    18# 2/4/3/8/    19# 4/1/2/6/    20# 4/1/3/7/
   2 13  8 11      2 16  5 11      2 11 14  7      2 11 14  7
   7 12  1 14      7  9  4 14     13  8  1 12     16  5  4  9
   9  6 15  4     12  6 15  1      3 10 15  6      3 10 15  6
  16  3 10  5     13  3 10  8     16  5  4  9     13  8  1 12
 21# 4/2/1/6/    22# 4/2/3/8/    23# 4/3/1/7/    24# 4/3/2/8/
   2 13 12  7      2 16  9  7      2 13 12  7      2 16  9  7
  11  8  1 14     11  5  4 14     16  3  6  9     13  3  6 12
   5 10 15  4      8 10 15  1      5 10 15  4      8 10 15  1
  16  3  6  9     13  3  6 12     11  8  1 14     11  5  4 14
 25# 1/4/6/2/    26# 1/4/7/3/    27# 2/4/6/1/    28# 2/4/8/3/
   3  6 15 10      3  6 15 10      3 13  8 10      3 16  5 10
  13 12  1  8     16  9  4  5      6 12  1 15      6  9  4 15
   2  7 14 11      2  7 14 11      9  7 14  4     12  7 14  1
  16  9  4  5     13 12  1  8     16  2 11  5     13  2 11  8
 29# 4/1/6/2/    30# 4/1/7/3/    31# 4/2/6/1/    32# 4/2/8/3/
   3 10 15  6      3 10 15  6      3 13 12  6      3 16  9  6
  13  8  1 12     16  5  4  9     10  8  1 15     10  5  4 15
   2 11 14  7      2 11 14  7      5 11 14  4      8 11 14  1
  16  5  4  9     13  8  1 12     16  2  7  9     13  2  7 12
```

```
 33# 4/3/7/1/    34# 4/3/8/2/    35# 1/4/6/7/    36# 1/4/7/6/
  3 13 12  6      3 16  9  6      4  5 16  9      4  5 16  9
 16  2  7  9     13  2  7 12     14 11  2  7     15 10  3  6
  5 11 14  4      8 11 14  1      1  8 13 12      1  8 13 12
 10  8  1 15     10  5  4 15     15 10  3  6     14 11  2  7
 37# 2/4/6/8/    38# 2/4/8/6/    39# 4/1/6/7/    40# 4/1/7/6/
  4 14  7  9      4 15  6  9      4  9 16  5      4  9 16  5
  5 11  2 16      5 10  3 16     14  7  2 11     15  6  3 10
 10  8 13  3     11  8 13  2      1 12 13  8      1 12 13  8
 15  1 12  6     14  1 12  7     15  6  3 10     14  7  2 11
 41# 4/2/6/8/    42# 4/2/8/6/    43# 4/3/7/8/    44# 4/3/8/7/
  4 14 11  5      4 15 10  5      4 14 11  5      4 15 10  5
  9  7  2 16      9  6  3 16     15  1  8 10     14  1  8 11
  6 12 13  3      7 12 13  2      6 12 13  3      7 12 13  2
 15  1  8 10     14  1  8 11      9  7  2 16      9  6  3 16
 45# 1/7/4/3/    46# 2/8/4/3/    47# 1/7/4/6/    48# 2/8/4/6/
  5  4 15 10      5 16  3 10      6  3 16  9      6 15  4  9
 16  9  6  3      4  9  6 15     15 10  5  4      3 10  5 16
  2  7 12 13     14  7 12  1      1  8 11 14     13  8 11  2
 11 14  1  8     11  2 13  8     12 13  2  7     12  1 14  7
[Count of Compositions = 48]
```

This list is formed by the new set of inequality conditions a little different from the former one: n1<n16; n1<n4; n4<n13;

Watch the 48 solutions carefully and find any correspondence among them. Pay your special attention to each number list of Used Units at first and try to find anything interesting.

[1] Among 12 'Standard Solutions' with n1=1 every list of used unit numbers consist of {1/, 2/, 3/ and 4/} only and nothing else.

It is necessary because the next proposition must be true among 4 layers of binary decompositions:

If V4n=8*An+4*Bn+2*Cn+Dn=0 then An=Bn=Cn=Dn=0;

n1=1 means n1=0 inside, and every layer must have n1=0;

You will find only each of {1/, 2/, 3/ and 4/} has n1=0 and nothing else.

[2] Compare these 12 'Standard Solutions' with the next 12 solutions of n1=2 piece by piece, you will find each pair of (1#, 13#),(2#, 14#),(3#, 15#),(4#, 16#), ..., (11#, 23#),(12#, 24#) has the first three numbers of used units always same with each other, and only the last one is different.

Unit number 1/ is always replaced by u8/; u2/ is replaced by u7/; and u3/ by u6/. No other pair could be found. What does this mean?

Compare the contents of Latin Units: (1/, 8/), (2/, 7/), (3/, 6/)

Could you find that each pair of them makes a 'Complementary Unit'?

Between those units each value of every corresponding position always add up to the constant value 1. If A+B=1 then we call (A, B) is a complementary pair of 1.

Yes. This is the 'Complementary Transformation' in unit level. You can get all the 12 solutions of n1=2 by transforming 12 'Standard Solutions' of n1=1 only replacing the last number of the used unit list: u1/ with u8/; u2/ with u7/; and u3/ with u6/.

. . . . . .

You may find any other interesting relations.

I myself made a new list of 3 groups of 16 solutions as shown below so that you

could easily see what I have found:

```
[Group 1]
   1# 1/2/4/3/     2# 2/4/3/1/     3# 4/2/1/3/     4# 4/3/1/2/
    1  8 11 14      1 15  6 12      1 14 11  8      1 14 11  8
   12 13  2  7      8 10  3 13     12  7  2 13     15  4  5 10
    6  3 16  9     11  5 16  2      6  9 16  3      6  9 16  3
   15 10  5  4     14  4  9  7     15  4  5 10     12  7  2 13
   5# 1/2/4/6/     6# 2/4/3/8/     7# 4/2/1/6/     8# 4/3/1/7/
    2  7 12 13      2 16  5 11      2 13 12  7      2 13 12  7
   11 14  1  8      7  9  4 14     11  8  1 14     16  3  6  9
    5  4 15 10     12  6 15  1      5 10 15  4      5 10 15  4
   16  9  6  3     13  3 10  8     16  3  6  9     11  8  1 14
   9# 1/7/4/3/    10# 2/4/6/1/    11# 4/2/8/3/    12# 4/3/8/2/
    5  4 15 10      3 13  8 10      3 16  9  6      3 16  9  6
   16  9  6  3      6 12  1 15     10  5  4 15     13  2  7 12
    2  7 12 13      9  7 14  4      8 11 14  1      8 11 14  1
   11 14  1  8     16  2 11  5     13  2  7 12     10  5  4 15
  13# 1/7/4/6/    14# 2/4/6/8/    15# 4/2/8/6/    16# 4/3/8/7/
    6  3 16  9      4 14  7  9      4 15 10  5      4 15 10  5
   15 10  5  4      5 11  2 16      9  6  3 16     14  1  8 11
    1  8 11 14     10  8 13  3      7 12 13  2      7 12 13  2
   12 13  2  7     15  1 12  6     14  1  8 11      9  6  3 16

[Group 2]
   1# 2/1/4/3/     2# 2/4/1/3/     3# 4/2/3/1/     4# 4/3/2/1/
    1 12  7 14      1 14  7 12      1 15 10  8      1 15 10  8
    8 13  2 11      8 11  2 13     12  6  3 13     14  4  5 11
   10  3 16  5     10  5 16  3      7  9 16  2      7  9 16  2
   15  6  9  4     15  4  9  6     14  4  5 11     12  6  3 13
   5# 2/1/4/6/     6# 2/4/1/6/     7# 4/2/3/8/     8# 4/3/2/8/
    2 11  8 13      2 13  8 11      2 16  9  7      2 16  9  7
    7 14  1 12      7 12  1 14     11  5  4 14     13  3  6 12
    9  4 15  6      9  6 15  4      8 10 15  1      8 10 15  1
   16  5 10  3     16  3 10  5     13  3  6 12     11  5  4 14
   9# 2/8/4/3/    10# 2/4/8/3/    11# 4/2/6/1/    12# 4/3/7/1/
    5 16  3 10      3 16  5 10      3 13 12  6      3 13 12  6
    4  9  6 15      6  9  4 15     10  8  1 15     16  2  7  9
   14  7 12  1     12  7 14  1      5 11 14  4      5 11 14  4
   11  2 13  8     13  2 11  8     16  2  7  9     10  8  1 15
  13# 2/8/4/6/    14# 2/4/8/6/    15# 4/2/6/8/    16# 4/3/7/8/
    6 15  4  9      4 15  6  9      4 14 11  5      4 14 11  5
    3 10  5 16      5 10  3 16      9  7  2 16     15  1  8 10
   13  8 11  2     11  8 13  2      6 12 13  3      6 12 13  3
   12  1 14  7     14  1 12  7     15  1  8 10      9  7  2 16

[Group 3]
   1# 1/4/2/3/     2# 1/4/3/2/     3# 4/1/2/3/     4# 4/1/3/2/
    1  8 13 12      1  8 13 12      1 12 13  8      1 12 13  8
   14 11  2  7     15 10  3  6     14  7  2 11     15  6  3 10
    4  5 16  9      4  5 16  9      4  9 16  5      4  9 16  5
   15 10  3  6     14 11  2  7     15  6  3 10     14  7  2 11
```

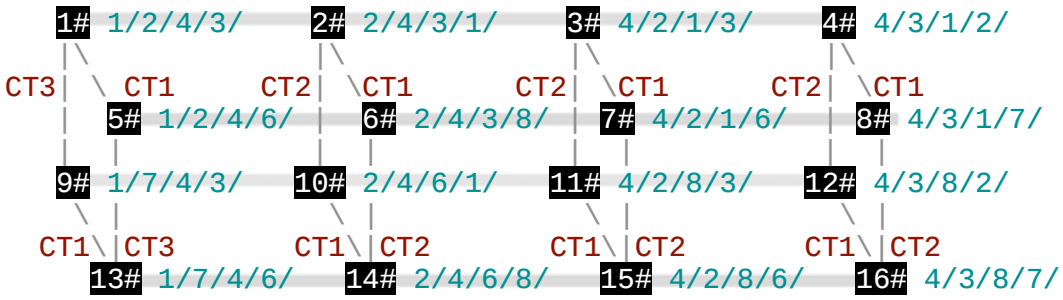
```
 5# 1/4/2/6/     6# 1/4/3/7/     7# 4/1/2/6/     8# 4/1/3/7/
 2  7 14 11      2  7 14 11      2 11 14  7      2 11 14  7
13 12  1  8     16  9  4  5     13  8  1 12     16  5  4  9
 3  6 15 10      3  6 15 10      3 10 15  6      3 10 15  6
16  9  4  5     13 12  1  8     16  5  4  9     13  8  1 12

 9# 1/4/7/3/    10# 1/4/6/2/    11# 4/1/7/3/    12# 4/1/6/2/
 3  6 15 10      3  6 15 10      3 10 15  6      3 10 15  6
16  9  4  5     13 12  1  8     16  5  4  9     13  8  1 12
 2  7 14 11      2  7 14 11      2 11 14  7      2 11 14  7
13 12  1  8     16  9  4  5     13  8  1 12     16  5  4  9

13# 1/4/7/6/    14# 1/4/6/7/    15# 4/1/7/6/    16# 4/1/6/7/
 4  5 16  9      4  5 16  9      4  9 16  5      4  9 16  5
15 10  3  6     14 11  2  7     15  6  3 10     14  7  2 11
 1  8 13 12      1  8 13 12      1 12 13  8      1 12 13  8
14 11  2  7     15 10  3  6     14  7  2 11     15  6  3 10
```

I have collected any solution which has the same contents in each row and each column into each group and made 3 groups of 16 solutions.

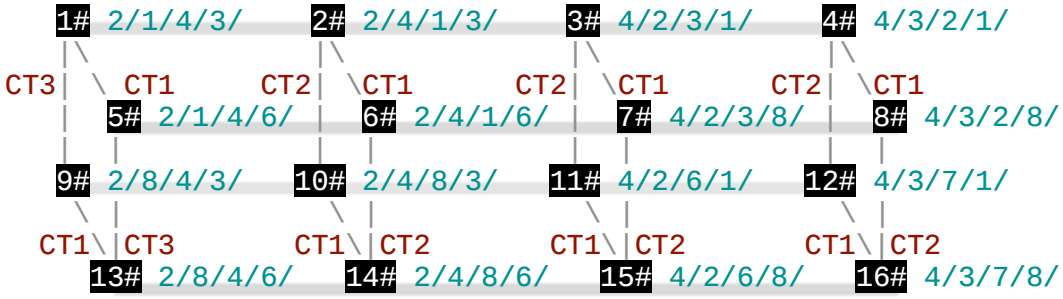
I have rearranged the positions of 16 solutions in each group as follows:

**** Complementary Transformations of Pan-diagonal MS44 ****

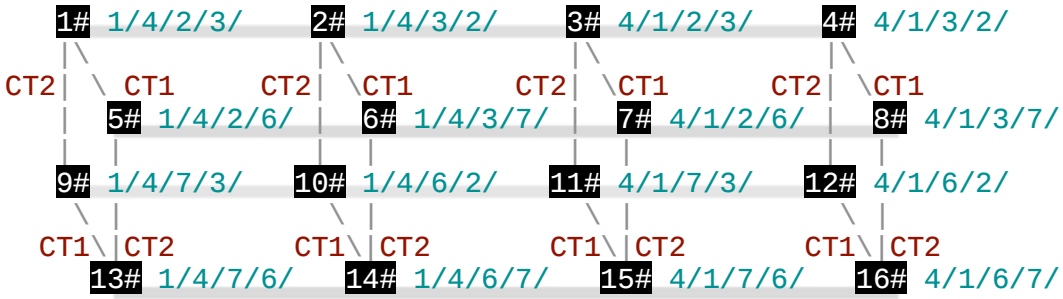
**[Group 1]**




**[Group 2]**




**[Group 3]**



The 3-dimensional figures above are drawn so as to show all the impressive correspondences between them. CT1 means 'Complementary Transformation' of the last numbers between the unit pair. CT2 means CT of the second numbers to the last. CT3 means the same of the third numbers to the last.

You may exchange those numbers of the used unit list to transform one into the other. How simple it is!

But that is not yet all for our job. We must find another simple rules for the 12 'Standard Solutions' of n1=1. How can we make all 12 from a few? What rules of exchange can we find among {1/, 2/, 3/ and 4/}?

But it was a difficult problem. I have not yet found anything simple and common, though I tried a lot to discover the satisfying ones. It took a long time in vain.

I decided to change my view-point at last. I stopped finding any more rules of transformation among them. I tried to study how those 12 solutions were made and listed out.

I checked the process itself exactly.

${}_4P_4$ of {1/, 2/, 3/, 4/} = 4x3x2x1 = 24. How were 12 'Standard Solutions chosen out of 24? I had to check it in the next list of 24 "Primitive Answers" below:

**** Pan-diagonal Magic Squares 4x4 Made by New Euler's Method ****
**[24 Pieces of "Primitive Answers"]**

```
  1/   2/   3/   4/          1#     1/   2/   4/   3/          2#
0011 0101 0101 0110   1  8 10 15   0011 0101 0110 0101   1  8 11 14
1100 0101 1010 1001  12 13  3  6   1100 0101 1001 1010  12 13  2  7
0011 1010 1010 0110   7  2 16  9   0011 1010 0110 1010   6  3 16  9
1100 1010 0101 1001  14 11  5  4   1100 1010 1001 0101  15 10  5  4

  1/   3/   2/   4/          3#     1/   3/   4/   2/          4#
0011 0101 0101 0110   1  8 10 15   0011 0101 0110 0101   1  8 11 14
1100 1010 0101 1001  14 11  5  4   1100 1010 1001 0101  15 10  5  4
0011 1010 1010 0110   7  2 16  9   0011 1010 0110 1010   6  3 16  9
1100 0101 1010 1001  12 13  3  6   1100 0101 1001 1010  12 13  2  7

  1/   4/   2/   3/          5#     1/   4/   3/   2/          6#
0011 0110 0101 0101   1  8 13 12   0011 0110 0101 0101   1  8 13 12
1100 1001 0101 1010  14 11  2  7   1100 1001 1010 0101  15 10  3  6
0011 0110 1010 1010   4  5 16  9   0011 0110 1010 1010   4  5 16  9
1100 1001 1010 0101  15 10  3  6   1100 1001 0101 1010  14 11  2  7

  2/   1/   3/   4/          7#     2/   1/   4/   3/          8#
0101 0011 0101 0110   1 12  6 15   0101 0011 0110 0101   1 12  7 14
0101 1100 1010 1001   8 13  3 10   0101 1100 1001 1010   8 13  2 11
1010 0011 1010 0110  11  2 16  5   1010 0011 0110 1010  10  3 16  5
1010 1100 0101 1001  14  7  9  4   1010 1100 1001 0101  15  6  9  4

  2/   3/   1/   4/          9#     2/   3/   4/   1/         10#
0101 0101 0011 0110   1 14  4 15   0101 0101 0110 0011   1 15  4 14
0101 1010 1100 1001   8 11  5 10   0101 1010 1001 1100   8 10  5 11
1010 1010 0011 0110  13  2 16  3   1010 1010 0110 0011  13  3 16  2
1010 0101 1100 1001  12  7  9  6   1010 0101 1001 1100  12  6  9  7

  2/   4/   1/   3/         11#     2/   4/   3/   1/         12#
0101 0110 0011 0101   1 14  7 12   0101 0110 0101 0011   1 15  6 12
0101 1001 1100 1010   8 11  2 13   0101 1001 1010 1100   8 10  3 13
1010 0110 0011 1010  10  5 16  3   1010 0110 1010 0011  11  5 16  2
1010 1001 1100 0101  15  4  9  6   1010 1001 0101 1100  14  4  9  7
```

```
  3/   1/   2/   4/          13#     3/   1/   4/   2/          14#
0101 0011 0101 0110   1 12  6 15   0101 0011 0110 0101   1 12  7 14
1010 1100 0101 1001  14  7  9  4   1010 1100 1001 0101  15  6  9  4
1010 0011 1010 0110  11  2 16  5   1010 0011 0110 1010  10  3 16  5
0101 1100 1010 1001   8 13  3 10   0101 1100 1001 1010   8 13  2 11
  3/   2/   1/   4/          15#     3/   2/   4/   1/          16#
0101 0101 0011 0110   1 14  4 15   0101 0101 0110 0011   1 15  4 14
1010 0101 1100 1001  12  7  9  6   1010 0101 1001 1100  12  6  9  7
1010 1010 0011 0110  13  2 16  3   1010 1010 0110 0011  13  3 16  2
0101 1010 1100 1001   8 11  5 10   0101 1010 1001 1100   8 10  5 11
  3/   4/   1/   2/          17#     3/   4/   2/   1/          18#
0101 0110 0011 0101   1 14  7 12   0101 0110 0101 0011   1 15  6 12
1010 1001 1100 0101  15  4  9  6   1010 1001 0101 1100  14  4  9  7
1010 0110 0011 1010  10  5 16  3   1010 0110 1010 0011  11  5 16  2
0101 1001 1100 1010   8 11  2 13   0101 1001 1010 1100   8 10  3 13
  4/   1/   2/   3/          19#     4/   1/   3/   2/          20#
0110 0011 0101 0101   1 12 13  8   0110 0011 0101 0101   1 12 13  8
1001 1100 0101 1010  14  7  2 11   1001 1100 1010 0101  15  6  3 10
0110 0011 1010 1010   4  9 16  5   0110 0011 1010 1010   4  9 16  5
1001 1100 1010 0101  15  6  3 10   1001 1100 0101 1010  14  7  2 11
  4/   2/   1/   3/          21#     4/   2/   3/   1/          22#
0110 0101 0011 0101   1 14 11  8   0110 0101 0101 0011   1 15 10  8
1001 0101 1100 1010  12  7  2 13   1001 0101 1010 1100  12  6  3 13
0110 1010 0011 1010   6  9 16  3   0110 1010 1010 0011   7  9 16  2
1001 1010 1100 0101  15  4  5 10   1001 1010 0101 1100  14  4  5 11
  4/   3/   1/   2/          23#     4/   3/   2/   1/          24#
0110 0101 0011 0101   1 14 11  8   0110 0101 0101 0011   1 15 10  8
1001 1010 1100 0101  15  4  5 10   1001 1010 0101 1100  14  4  5 11
0110 1010 0011 1010   6  9 16  3   0110 1010 1010 0011   7  9 16  2
1001 0101 1100 1010  12  7  2 13   1001 0101 1010 1100  12  6  3 13
```

It is the inequality condition n4<n13 that chooses 12 out of 24. It strictly cuts off all with n4 on the right top corner greater than n13 on the left bottom without any exception.

Then how can we make it into the rules of creating 12 'Standard Solutions'?

We have to know at first what conditions correspond to this selective n4<n13, first in the Latin Units and second in the number lists of used units.

[Latin Units]

```
     1/        2/        3/        4/
 0 0 1 1   0 1 0 1   0 1 0 1   0 1 1 0
 1 1 0 0   0 1 0 1   1 0 1 0   1 0 0 1
 0 0 1 1   1 0 1 0   1 0 1 0   0 1 1 0
 1 1 0 0   1 0 1 0   0 1 0 1   1 0 0 1
```

No.3/ and 4/ units are only active to determine which is greater in total, either n4 or n13.

In the number list of used units No.3/ must not be put in any position upper than No.4/ unit, no matter where they are. Otherwise the value of n4 should be greater than n13 in total.

Therefore No.3/ must not be put at the top position, and No.4/ must not be put at the bottom, either.

We can use these rules for our purpose, can't we?

[1] Make permutations taking 4 out of {1/, 2/, 3/ and 4} by simple change of contents order of number list as below:

$_4P_4$ =

{1/,2/,3/,4/}, {1/,2/,4/,3/}, {1/,3/,2/,4/}, {1/,3/,4/,2/},
{1/,4/,2/,3/}, {1/,4/,3/,2/}, {2/,1/,3/,4/}, {2/,1/,4/,3/},
{2/,3/,1/,4/}, {2/,3/,4/,1/}, {2/,4/,1/,3/}, {2/,4/,3/,1/},
. . . . .
{4/,2/,1/,3/}, {4/,2/,3/,1/}, {4/,3/,1/,2/}, {4/,3/,2/,1/}

[2] Select only those which have 4/ upper than 3/ in their list positions, and eliminate all with 3/ upper than 4/.

You can surely get the very 12 'Standard Solutions' of n1=1 you want.

[3] Recompose all the 'Standard Solutions' by the 4 Latin Units and the number lists of used units, and classify them into 3 groups according the contents of each row and each column. Then you can finally have 4 Standard Solutions in each group.

What do you think of these rules? I am afraid they may not look very beautiful, but I believe they are enough to explain how those 12 are created.

If you have only Latin Units and number list of used units, you can explain everything analytically by them. How smart it is!

I hope new style of analytical study will start and grow, and finally produce a lot of good results in any type and in any order.

### 3. About "Non-Complete-Euler-Squares" Again

When we are studying only "Complete Euler Squares", we often tend to think that there are only those types in this world. But it is not true, especially in the case of higher orders. 'Complete Euler Squares' really make the minority in the world of orders 7, 8 and 9. There are many "Non-Complete-Euler-Squares" here, far more than Complete Euler Squares.

Take your kind look at the next examples as listed below.

These are some examples of 'Complete' pan-magic squares of order 8. Each of them has all 32 complementary pairs of 65 located only on its pan-diagonals.

**** 'Complete' Pan-magic Squares 8x8 with Binary Decompositions ****

| | 1# | /D2i 32* | 16* | 8* | 4* | 2* | 1* |
|---|---|---|---|---|---|---|---|
| 1 62 5 59 2 61 12 58 | | 01010101 | 01010101 | 01010111 | 01100100 | 00010010 | 01001011 |
| 57 14 50 48 9 18 45 19 | | 10110010 | 10100101 | 11011010 | 01010010 | 00010001 | 01110100 |
| 10 27 34 25 54 33 36 41 | | 00101111 | 01011000 | 11010001 | 00001000 | 01000010 | 10101010 |
| 49 30 26 23 52 21 22 37 | | 10001001 | 11111110 | 01100000 | 01010111 | 00011000 | 01101010 |
| 63 4 53 7 64 3 60 6 | | 10101010 | 10101010 | 10001010 | 10111001 | 11011110 | 01001011 |
| 56 47 20 46 8 51 15 17 | | 11010100 | 10100101 | 01010010 | 11011010 | 11101110 | 10111000 |
| 11 32 29 24 55 38 31 40 | | 00001101 | 01111010 | 11100010 | 01111111 | 11011011 | 01010101 |
| 13 44 43 28 16 35 39 42 | | 01100111 | 00010000 | 11111001 | 10001010 | 01111110 | 01011001 |
| | 9# | /D2i 32* | 16* | 8* | 4* | 2* | 1* |
| 1 62 5 59 2 61 12 58 | | 01010101 | 01010101 | 01010111 | 01100100 | 00010010 | 01001011 |
| 57 14 50 48 9 18 45 19 | | 10110010 | 10100101 | 11011010 | 01010010 | 00010001 | 01110100 |
| 10 28 35 25 54 33 36 39 | | 00101111 | 01011000 | 11010000 | 00001001 | 01100011 | 11001010 |
| 49 31 24 22 52 23 21 38 | | 10001001 | 11111110 | 01000000 | 01110111 | 01101100 | 00111001 |
| 63 4 53 7 64 3 60 6 | | 10101010 | 10101010 | 10001010 | 10111001 | 11011110 | 01001011 |
| 56 47 20 46 8 51 15 17 | | 11010100 | 10100101 | 01010010 | 11011010 | 11101110 | 10111000 |
| 11 32 29 26 55 37 30 40 | | 00001101 | 01111010 | 11110010 | 01101111 | 11001001 | 01010011 |
| 13 42 44 27 16 34 41 43 | | 01100111 | 00010000 | 11111011 | 10001000 | 00111001 | 01101100 |

```
                      17#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 29 32 25 54 28 38 44  00001011 01111100 11110101 01101010 00100101 10101111
49 26 30 23 52 24 22 34  10001001 11111110 01100000 00110110 00011100 01101111
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 37 27 21 55 36 33 40  01001111 00111000 10100000 01011001 10101101 00000101
13 41 43 31 16 39 35 42  01100111 00010000 11111001 10011100 00111110 00001001
                      33#  /D2i 32*      16*       8*       4*       2*       1*
 1 62  5 59  2 61 12 58  01010101 01010101 01010111 01100100 00010010 01001011
57 14 50 48  9 18 45 19  10110010 10100101 11011010 01010010 00010001 01110100
10 30 34 22 54 28 38 44  00101011 01011100 11000101 01011010 00000101 11111111
49 26 29 24 52 25 23 32  10001000 11111111 01100101 00110011 00011011 01011001
63  4 53  7 64  3 60  6  10101010 10101010 10001010 10111001 11011110 01001011
56 47 20 46  8 51 15 17  11010100 10100101 01010010 11011010 11101110 10111000
11 37 27 21 55 35 31 43  01001101 00111010 10100011 01011010 10101111 00000000
13 40 42 33 16 39 36 41  01110111 00000000 10101001 11001100 01001110 01101010
. . . . . .
```

Therefore each pan-diagonal consists of 4 complementary pairs of 65 and adds up to the magic constant 260. Yes. They are one of the special types of Pan-diagonal MS88. Each of them corresponds to each of Self-complementary MS88 by one to one.

Watch their contents of binary decompositions above. Each row or each column in every layer is not always made up of four '0' and four '1'. It does not always have the very number pattern of 'Complete Euler Square' {0,0,0,0,1,1,1,1}, though most pan-diagonals seem to obey the standard.

Every Non-Euler type does not always accept our presumption that each layer must have the same properties in common, and each row and each column must be made of four '0' and four '1' and add up to the same sum even in every layer.

But they are surely 'Complete' MS88 on the whole and each row, each column and each pan-diagonal adds up to the constant sum in total.

Compare those to the next examples of 'Complete Euler Squares'.

**** 'Complete Euler Squares' 8x8 with Binary Decompositions ****

```
                       1#  /D2i 32*      16*       8*       4*       2*       1*
 1 63  6 60 14 52  9 55  01010101 01010101 01011010 01101001 01010101 00111100
62  4 57  7 49 15 54 12  10101010 10101010 10100101 10010110 01010101 11000011
19 45 24 42 32 34 27 37  01010101 10101010 01011010 01101001 10101010 00111100
48 18 43 21 35 29 40 26  10101010 01010101 10100101 10010110 10101010 11000011
51 13 56 10 64  2 59  5  10101010 10101010 01011010 01101001 10101010 00111100
16 50 11 53  3 61  8 58  01010101 01010101 10100101 10010110 10101010 11000011
33 31 38 28 46 20 41 23  10101010 01010101 01011010 01101001 01010101 00111100
30 36 25 39 17 47 22 44  01010101 10101010 10100101 10010110 01010101 11000011
                       2#  /D2i 32*      16*       8*       4*       2*       1*
 1 63  6 60 22 44 17 47  01010101 01011010 01010101 01101001 01010101 00111100
62  4 57  7 41 23 46 20  10101010 10100101 10101010 10010110 01010101 11000011
11 53 16 50 32 34 27 37  01010101 01011010 10101010 01101001 10101010 00111100
56 10 51 13 35 29 40 26  10101010 10100101 01010101 10010110 10101010 11000011
43 21 48 18 64  2 59  5  10101010 01011010 10101010 01101001 10101010 00111100
24 42 19 45  3 61  8 58  01010101 10100101 01010101 10010110 10101010 11000011
33 31 38 28 54 12 49 15  10101010 01011010 01010101 01101001 01010101 00111100
30 36 25 39  9 55 14 52  01010101 10100101 10101010 10010110 01010101 11000011
```

```
                        3#  /D2i 32*      16*       8*       4*       2*       1*
  1 63  6 60 38 28 33 31  01011010 01010101 01010101 01101001 01010101 00111100
 62  4 57  7 25 39 30 36  10100101 10101010 10101010 10010110 01010101 11000011
 11 53 16 50 48 18 43 21  01011010 01010101 10101010 01101001 10101010 00111100
 56 10 51 13 19 45 24 42  10100101 10101010 01010101 10010110 10101010 11000011
 27 37 32 34 64  2 59  5  01011010 10101010 10101010 01101001 10101010 00111100
 40 26 35 29  3 61  8 58  10100101 01010101 01010101 10010110 10101010 11000011
 17 47 22 44 54 12 49 15  01011010 10101010 01010101 01101001 01010101 00111100
 46 20 41 23  9 55 14 52  10100101 01010101 10101010 10010110 01010101 11000011
```

How fine their binary decompositions are!

But they exist quite rare in this world, and they are shining brightly just like precious jewels.

I would like to think again about the reason why Non-C.Euler type of higher orders could exist so many.

In the former section we know only {0,0,0,0,1,1,1,1} should be possible for any line in each layer of binary decompositions of Euler Squares 8x8.

But if it is true and enough, you might think everything should be of Complete Euler type, and nothing could be of Non-C.Euler type. But it is neither true nor enough in reality. What makes so many Non-C.E.S. in this world?

Another explanation has to be made for it.

Non-C.E.S. do not always have to accept the strict standard of Euler Squares, while each row, each column and each pan-diagonal adds up to the magic constant in total. They don't care anything of their binary decompositions, but they only care of the whole conditions.

Then what are possible in every layer of binary decompositions, while each line adds up to the same sum on the whole? We have to check what value patterns could appear in each binary layer, in every row, every column and every pan-diagonal.

$$V_8n = 32*An+16*Bn+8*Cn+4*Dn+2*En+1*Fn = Sn \quad (n=1,2,3,4, \ldots , 63,64)$$

Calculation must be made under the equation above. Sn = 252(Constant);

Suppose An, Bn, Cn, ..., Fn mean any possible sums of each row, each column and each pan-diagonal in each layer. Each of {An, Bn, Cn, ..., Fn} must take any one of values {0, 1, 2, 3, 4, 5, 6, 7 and 8}. Two or more variables may take the same value.

What value patterns are possible for {An, Bn, Cn, ..., Fn}?

The next list shows the result of my recent calculation for every case of every order:

```
 ** Possible Value Patterns for any Line-Sum in
  each Layer Adding up to the Equal Sum on the Whole **

 [Order 3; Base 3]
If [3*An+Bn==12] then [{An,Bn} = {2,6}, {3,3}, {4,0}]
* Count = 3

 [Order 4; Base 2]
If [8*An+4*Bn+2*Cn+Dn==30] then [{An,Bn,Cn,Dn} =
  {1,3,3,4},{1,3,4,2},{1,4,1,4},{1,4,2,2},{1,4,3,0},
  {2,1,3,4},{2,1,4,2},{2,2,1,4},{2,2,2,2},{2,2,3,0},{2,3,0,2},{2,3,1,0},
  {3,0,1,4},{3,0,2,2},{3,0,3,0},{3,1,0,2},{3,1,1,0}]
* Count = 17
```

```
 [Order 5; Base 5]
If [5*An+Bn==60] then [{An,Bn} =
  {8,20}, {9,15}, {10,10}, {11,5}, {12,0}]
* Count = 5

 [Order 7; Base 7]
If [7*An+Bn==168] then [{An,Bn} =
  {18,42}, {19,35}, {20,28}, {21,21}, {22,14}, {23,7}, {24,0}]
* Count = 7

 [Order 8; Base 2]
If [32*An+16*Bn+8*Cn+4*Dn+2*En+Fn==252] then [{An,Bn,Cn,Dn,En,Fn} =
  {1,7,7,7,8,8}, {1,7,7,8,6,8}, {1,7,7,8,7,6}, {1,7,7,8,8,4}, {1,7,8,5,8,8},
  {1,7,8,6,6,8}, {1,7,8,6,7,6}, {1,7,8,6,8,4}, {1,7,8,7,4,8}, {1,7,8,7,5,6},
  {1,7,8,7,6,4}, {1,7,8,7,7,2}, {1,7,8,7,8,0}, {1,7,8,8,2,8}, {1,7,8,8,3,6},
  {1,7,8,8,4,4}, {1,7,8,8,5,2}, {1,7,8,8,6,0}, {1,8,5,7,8,8}, {1,8,5,8,6,8},
  {1,8,5,8,7,6}, {1,8,5,8,8,4}, {1,8,6,5,8,8}, {1,8,6,6,6,8}, {1,8,6,6,7,6},
  {1,8,6,6,8,4}, {1,8,6,7,4,8}, {1,8,6,7,5,6}, {1,8,6,7,6,4}, {1,8,6,7,7,2},
  {1,8,6,7,8,0}, {1,8,6,8,2,8}, {1,8,6,8,3,6}, {1,8,6,8,4,4}, {1,8,6,8,5,2},
  {1,8,6,8,6,0}, {1,8,7,3,8,8}, {1,8,7,4,6,8}, {1,8,7,4,7,6}, {1,8,7,4,8,4},
  {1,8,7,5,4,8}, {1,8,7,5,5,6}, {1,8,7,5,6,4}, {1,8,7,5,7,2}, {1,8,7,5,8,0},
  {1,8,7,6,2,8}, {1,8,7,6,3,6}, {1,8,7,6,4,4}, {1,8,7,6,5,2}, {1,8,7,6,6,0},
  {1,8,7,7,0,8}, {1,8,7,7,1,6}, {1,8,7,7,2,4}, {1,8,7,7,3,2}, {1,8,7,7,4,0},
  {1,8,7,8,0,4}, {1,8,7,8,1,2}, {1,8,7,8,2,0}, {1,8,8,1,8,8}, {1,8,8,2,6,8},
  {1,8,8,2,7,6}, {1,8,8,2,8,4}, {1,8,8,3,4,8}, {1,8,8,3,5,6}, {1,8,8,3,6,4},
  {1,8,8,3,7,2}, {1,8,8,3,8,0}, {1,8,8,4,2,8}, {1,8,8,4,3,6}, {1,8,8,4,4,4},
  {1,8,8,4,5,2}, {1,8,8,4,6,0}, {1,8,8,5,0,8}, {1,8,8,5,1,6}, {1,8,8,5,2,4},
  {1,8,8,5,3,2}, {1,8,8,5,4,0}, {1,8,8,6,0,4}, {1,8,8,6,1,2}, {1,8,8,6,2,0},
  {1,8,8,7,0,0}, {2,5,7,7,8,8}, {2,5,7,8,6,8}, {2,5,7,8,7,6}, {2,5,7,8,8,4},
  {2,5,8,5,8,8}, {2,5,8,6,6,8}, {2,5,8,6,7,6}, {2,5,8,6,8,4}, {2,5,8,7,4,8},

  {3,8,3,0,0,4}, {3,8,3,0,1,2}, {3,8,3,0,2,0}, {3,8,3,1,0,0}, {4,1,7,7,8,8},
  {4,1,7,8,6,8}, {4,1,7,8,7,6}, {4,1,7,8,8,4}, {4,1,8,5,8,8}, {4,1,8,6,6,8},
  {4,1,8,6,7,6}, {4,1,8,6,8,4}, {4,1,8,7,4,8}, {4,1,8,7,5,6}, {4,1,8,7,6,4},
  {4,1,8,7,7,2}, {4,1,8,7,8,0}, {4,1,8,8,2,8}, {4,1,8,8,3,6}, {4,1,8,8,4,4},
  {4,1,8,8,5,2}, {4,1,8,8,6,0}, {4,2,5,7,8,8}, {4,2,5,8,6,8}, {4,2,5,8,7,6},
  {4,2,5,8,8,4}, {4,2,6,5,8,8}, {4,2,6,6,6,8}, {4,2,6,6,7,6}, {4,2,6,6,8,4},
  {4,2,6,7,4,8}, {4,2,6,7,5,6}, {4,2,6,7,6,4}, {4,2,6,7,7,2}, {4,2,6,7,8,0},
  {4,2,6,8,2,8}, {4,2,6,8,3,6}, {4,2,6,8,4,4}, {4,2,6,8,5,2}, {4,2,6,8,6,0},
  {4,2,7,3,8,8}, {4,2,7,4,6,8}, {4,2,7,4,7,6}, {4,2,7,4,8,4}, {4,2,7,5,4,8},
  {4,2,7,5,5,6}, {4,2,7,5,6,4}, {4,2,7,5,7,2}, {4,2,7,5,8,0}, {4,2,7,6,2,8},
  {4,2,7,6,3,6}, {4,2,7,6,4,4}, {4,2,7,6,5,2}, {4,2,7,6,6,0}, {4,2,7,7,0,8},
  {4,2,7,7,1,6}, {4,2,7,7,2,4}, {4,2,7,7,3,2}, {4,2,7,7,4,0}, {4,2,7,8,0,4},
  {4,2,7,8,1,2}, {4,2,7,8,2,0}, {4,2,8,1,8,8}, {4,2,8,2,6,8}, {4,2,8,2,7,6},
  {4,2,8,2,8,4}, {4,2,8,3,4,8}, {4,2,8,3,5,6}, {4,2,8,3,6,4}, {4,2,8,3,7,2},
  {4,2,8,3,8,0}, {4,2,8,4,2,8}, {4,2,8,4,3,6}, {4,2,8,4,4,4}, {4,2,8,4,5,2},
  {4,2,8,4,6,0}, {4,2,8,5,0,8}, {4,2,8,5,1,6}, {4,2,8,5,2,4}, {4,2,8,5,3,2},
  {4,2,8,5,4,0}, {4,2,8,6,0,4}, {4,2,8,6,1,2}, {4,2,8,6,2,0}, {4,2,8,7,0,0},
  {4,3,3,7,8,8}, {4,3,3,8,6,8}, {4,3,3,8,7,6}, {4,3,3,8,8,4}, {4,3,4,5,8,8},
  {4,3,4,6,6,8}, {4,3,4,6,7,6}, {4,3,4,6,8,4}, {4,3,4,7,4,8}, {4,3,4,7,5,6},
  {4,3,4,7,6,4}, {4,3,4,7,7,2}, {4,3,4,7,8,0}, {4,3,4,8,2,8}, {4,3,4,8,3,6},
  {4,3,4,8,4,4}, {4,3,4,8,5,2}, {4,3,4,8,6,0}, {4,3,5,3,8,8}, {4,3,5,4,6,8},
  {4,3,5,4,7,6}, {4,3,5,4,8,4}, {4,3,5,5,4,8}, {4,3,5,5,5,6}, {4,3,5,5,6,4},
  {4,3,5,5,7,2}, {4,3,5,5,8,0}, {4,3,5,6,2,8}, {4,3,5,6,3,6}, {4,3,5,6,4,4},
  {4,3,5,6,5,2}, {4,3,5,6,6,0}, {4,3,5,7,0,8}, {4,3,5,7,1,6}, {4,3,5,7,2,4},
  {4,3,5,7,3,2}, {4,3,5,7,4,0}, {4,3,5,8,0,4}, {4,3,5,8,1,2}, {4,3,5,8,2,0},
  {4,3,6,1,8,8}, {4,3,6,2,6,8}, {4,3,6,2,7,6}, {4,3,6,2,8,4}, {4,3,6,3,4,8},
```

{4,3,6,3,5,6}, {4,3,6,3,6,4}, {4,3,6,3,7,2}, {4,3,6,3,8,0}, {4,3,6,4,2,8},
{4,3,6,4,3,6}, {4,3,6,4,4,4}, {4,3,6,4,5,2}, {4,3,6,4,6,0}, {4,3,6,5,0,8},
{4,3,6,5,1,6}, {4,3,6,5,2,4}, {4,3,6,5,3,2}, {4,3,6,5,4,0}, {4,3,6,6,0,4},
{4,3,6,6,1,2}, {4,3,6,6,2,0}, {4,3,6,7,0,0}, {4,3,7,0,6,8}, {4,3,7,0,7,6},
{4,3,7,0,8,4}, {4,3,7,1,4,8}, {4,3,7,1,5,6}, {4,3,7,1,6,4}, {4,3,7,1,7,2},
{4,3,7,1,8,0}, {4,3,7,2,2,8}, {4,3,7,2,3,6}, {4,3,7,2,4,4}, {4,3,7,2,5,2},
{4,3,7,2,6,0}, {4,3,7,3,0,8}, {4,3,7,3,1,6}, {4,3,7,3,2,4}, {4,3,7,3,3,2},
{4,3,7,3,4,0}, {4,3,7,4,0,4}, {4,3,7,4,1,2}, {4,3,7,4,2,0}, {4,3,7,5,0,0},
{4,3,8,0,2,8}, {4,3,8,0,3,6}, {4,3,8,0,4,4}, {4,3,8,0,5,2}, {4,3,8,0,6,0},
{4,3,8,1,0,8}, {4,3,8,1,1,6}, {4,3,8,1,2,4}, {4,3,8,1,3,2}, {4,3,8,1,4,0},
{4,3,8,2,0,4}, {4,3,8,2,1,2}, {4,3,8,2,2,0}, {4,3,8,3,0,0}, {4,4,1,7,8,8},
{4,4,1,8,6,8}, {4,4,1,8,7,6}, {4,4,1,8,8,4}, {4,4,2,5,8,8}, {4,4,2,6,6,8},
{4,4,2,6,7,6}, {4,4,2,6,8,4}, {4,4,2,7,4,8}, {4,4,2,7,5,6}, {4,4,2,7,6,4},
{4,4,2,7,7,2}, {4,4,2,7,8,0}, {4,4,2,8,2,8}, {4,4,2,8,3,6}, {4,4,2,8,4,4},
{4,4,2,8,5,2}, {4,4,2,8,6,0}, {4,4,3,3,8,8}, {4,4,3,4,6,8}, {4,4,3,4,7,6},
{4,4,3,4,8,4}, {4,4,3,5,4,8}, {4,4,3,5,5,6}, {4,4,3,5,6,4}, {4,4,3,5,7,2},
{4,4,3,5,8,0}, {4,4,3,6,2,8}, {4,4,3,6,3,6}, {4,4,3,6,4,4}, {4,4,3,6,5,2},
{4,4,3,6,6,0}, {4,4,3,7,0,8}, {4,4,3,7,1,6}, {4,4,3,7,2,4}, {4,4,3,7,3,2},
{4,4,3,7,4,0}, {4,4,3,8,0,4}, {4,4,3,8,1,2}, {4,4,3,8,2,0}, {4,4,4,1,8,8},
{4,4,4,2,6,8}, {4,4,4,2,7,6}, {4,4,4,2,8,4}, {4,4,4,3,4,8}, {4,4,4,3,5,6},
{4,4,4,3,6,4}, {4,4,4,3,7,2}, {4,4,4,3,8,0}, {4,4,4,4,2,8}, {4,4,4,4,3,6},
{**4,4,4,4,4,4**}, {4,4,4,4,5,2}, {4,4,4,4,6,0}, {4,4,4,5,0,8}, {4,4,4,5,1,6},
{4,4,4,5,2,4}, {4,4,4,5,3,2}, {4,4,4,5,4,0}, {4,4,4,6,0,4}, {4,4,4,6,1,2},
{4,4,4,6,2,0}, {4,4,4,7,0,0}, {4,4,5,0,6,8}, {4,4,5,0,7,6}, {4,4,5,0,8,4},
{4,4,5,1,4,8}, {4,4,5,1,5,6}, {4,4,5,1,6,4}, {4,4,5,1,7,2}, {4,4,5,1,8,0},
{4,4,5,2,2,8}, {4,4,5,2,3,6}, {4,4,5,2,4,4}, {4,4,5,2,5,2}, {4,4,5,2,6,0},
{4,4,5,3,0,8}, {4,4,5,3,1,6}, {4,4,5,3,2,4}, {4,4,5,3,3,2}, {4,4,5,3,4,0},
{4,4,5,4,0,4}, {4,4,5,4,1,2}, {4,4,5,4,2,0}, {4,4,5,5,0,0}, {4,4,6,0,2,8},
{4,4,6,0,3,6}, {4,4,6,0,4,4}, {4,4,6,0,5,2}, {4,4,6,0,6,0}, {4,4,6,1,0,8},
{4,4,6,1,1,6}, {4,4,6,1,2,4}, {4,4,6,1,3,2}, {4,4,6,1,4,0}, {4,4,6,2,0,4},
{4,4,6,2,1,2}, {4,4,6,2,2,0}, {4,4,6,3,0,0}, {4,4,7,0,0,4}, {4,4,7,0,1,2},
{4,4,7,0,2,0}, {4,4,7,1,0,0}, {4,5,0,5,8,8}, {4,5,0,6,6,8}, {4,5,0,6,7,6},
{4,5,0,6,8,4}, {4,5,0,7,4,8}, {4,5,0,7,5,6}, {4,5,0,7,6,4}, {4,5,0,7,7,2},
{4,5,0,7,8,0}, {4,5,0,8,2,8}, {4,5,0,8,3,6}, {4,5,0,8,4,4}, {4,5,0,8,5,2},
{4,5,0,8,6,0}, {4,5,1,3,8,8}, {4,5,1,4,6,8}, {4,5,1,4,7,6}, {4,5,1,4,8,4},
{4,5,1,5,4,8}, {4,5,1,5,5,6}, {4,5,1,5,6,4}, {4,5,1,5,7,2}, {4,5,1,5,8,0},
{4,5,1,6,2,8}, {4,5,1,6,3,6}, {4,5,1,6,4,4}, {4,5,1,6,5,2}, {4,5,1,6,6,0},
{4,5,1,7,0,8}, {4,5,1,7,1,6}, {4,5,1,7,2,4}, {4,5,1,7,3,2}, {4,5,1,7,4,0},
{4,5,1,8,0,4}, {4,5,1,8,1,2}, {4,5,1,8,2,0}, {4,5,2,1,8,8}, {4,5,2,2,6,8},
{4,5,2,2,7,6}, {4,5,2,2,8,4}, {4,5,2,3,4,8}, {4,5,2,3,5,6}, {4,5,2,3,6,4},
{4,5,2,3,7,2}, {4,5,2,3,8,0}, {4,5,2,4,2,8}, {4,5,2,4,3,6}, {4,5,2,4,4,4},
{4,5,2,4,5,2}, {4,5,2,4,6,0}, {4,5,2,5,0,8}, {4,5,2,5,1,6}, {4,5,2,5,2,4},
{4,5,2,5,3,2}, {4,5,2,5,4,0}, {4,5,2,6,0,4}, {4,5,2,6,1,2}, {4,5,2,6,2,0},
{4,5,2,7,0,0}, {4,5,3,0,6,8}, {4,5,3,0,7,6}, {4,5,3,0,8,4}, {4,5,3,1,4,8},
{4,5,3,1,5,6}, {4,5,3,1,6,4}, {4,5,3,1,7,2}, {4,5,3,1,8,0}, {4,5,3,2,2,8},
{4,5,3,2,3,6}, {4,5,3,2,4,4}, {4,5,3,2,5,2}, {4,5,3,2,6,0}, {4,5,3,3,0,8},
{4,5,3,3,1,6}, {4,5,3,3,2,4}, {4,5,3,3,3,2}, {4,5,3,3,4,0}, {4,5,3,4,0,4},
{4,5,3,4,1,2}, {4,5,3,4,2,0}, {4,5,3,5,0,0}, {4,5,4,0,2,8}, {4,5,4,0,3,6},
{4,5,4,0,4,4}, {4,5,4,0,5,2}, {4,5,4,0,6,0}, {4,5,4,1,0,8}, {4,5,4,1,1,6},
{4,5,4,1,2,4}, {4,5,4,1,3,2}, {4,5,4,1,4,0}, {4,5,4,2,0,4}, {4,5,4,2,1,2},
{4,5,4,2,2,0}, {4,5,4,3,0,0}, {4,5,5,0,0,4}, {4,5,5,0,1,2}, {4,5,5,0,2,0},
{4,5,5,1,0,0}, {4,6,0,1,8,8}, {4,6,0,2,6,8}, {4,6,0,2,7,6}, {4,6,0,2,8,4},
{4,6,0,3,4,8}, {4,6,0,3,5,6}, {4,6,0,3,6,4}, {4,6,0,3,7,2}, {4,6,0,3,8,0},
{4,6,0,4,2,8}, {4,6,0,4,3,6}, {4,6,0,4,4,4}, {4,6,0,4,5,2}, {4,6,0,4,6,0},
{4,6,0,5,0,8}, {4,6,0,5,1,6}, {4,6,0,5,2,4}, {4,6,0,5,3,2}, {4,6,0,5,4,0},
{4,6,0,6,0,4}, {4,6,0,6,1,2}, {4,6,0,6,2,0}, {4,6,0,7,0,0}, {4,6,1,0,6,8},
{4,6,1,0,7,6}, {4,6,1,0,8,4}, {4,6,1,1,4,8}, {4,6,1,1,5,6}, {4,6,1,1,6,4},
{4,6,1,1,7,2}, {4,6,1,1,8,0}, {4,6,1,2,2,8}, {4,6,1,2,3,6}, {4,6,1,2,4,4},
{4,6,1,2,5,2}, {4,6,1,2,6,0}, {4,6,1,3,0,8}, {4,6,1,3,1,6}, {4,6,1,3,2,4},

```
  {4,6,1,3,3,2}, {4,6,1,3,4,0}, {4,6,1,4,0,4}, {4,6,1,4,1,2}, {4,6,1,4,2,0},
  {4,6,1,5,0,0}, {4,6,2,0,2,8}, {4,6,2,0,3,6}, {4,6,2,0,4,4}, {4,6,2,0,5,2},
  {4,6,2,0,6,0}, {4,6,2,1,0,8}, {4,6,2,1,1,6}, {4,6,2,1,2,4}, {4,6,2,1,3,2},
  {4,6,2,1,4,0}, {4,6,2,2,0,4}, {4,6,2,2,1,2}, {4,6,2,2,2,0}, {4,6,2,3,0,0},
  {4,6,3,0,0,4}, {4,6,3,0,1,2}, {4,6,3,0,2,0}, {4,6,3,1,0,0}, {4,7,0,0,2,8},
  {4,7,0,0,3,6}, {4,7,0,0,4,4}, {4,7,0,0,5,2}, {4,7,0,0,6,0}, {4,7,0,1,0,8},
  {4,7,0,1,1,6}, {4,7,0,1,2,4}, {4,7,0,1,3,2}, {4,7,0,1,4,0}, {4,7,0,2,0,4},
  {4,7,0,2,1,2}, {4,7,0,2,2,0}, {4,7,0,3,0,0}, {4,7,1,0,0,4}, {4,7,1,0,1,2},
  {4,7,1,0,2,0}, {4,7,1,1,0,0}, {5,0,5,7,8,8}, {5,0,5,8,6,8}, {5,0,5,8,7,6},

  {7,0,0,1,8,8}, {7,0,0,2,6,8}, {7,0,0,2,7,6}, {7,0,0,2,8,4}, {7,0,0,3,4,8},
  {7,0,0,3,5,6}, {7,0,0,3,6,4}, {7,0,0,3,7,2}, {7,0,0,3,8,0}, {7,0,0,4,2,8},
  {7,0,0,4,3,6}, {7,0,0,4,4,4}, {7,0,0,4,5,2}, {7,0,0,4,6,0}, {7,0,0,5,0,8},
  {7,0,0,5,1,6}, {7,0,0,5,2,4}, {7,0,0,5,3,2}, {7,0,0,5,4,0}, {7,0,0,6,0,4},
  {7,0,0,6,1,2}, {7,0,0,6,2,0}, {7,0,0,7,0,0}, {7,0,1,0,6,8}, {7,0,1,0,7,6},
  {7,0,1,0,8,4}, {7,0,1,1,4,8}, {7,0,1,1,5,6}, {7,0,1,1,6,4}, {7,0,1,1,7,2},
  {7,0,1,1,8,0}, {7,0,1,2,2,8}, {7,0,1,2,3,6}, {7,0,1,2,4,4}, {7,0,1,2,5,2},
  {7,0,1,2,6,0}, {7,0,1,3,0,8}, {7,0,1,3,1,6}, {7,0,1,3,2,4}, {7,0,1,3,3,2},
  {7,0,1,3,4,0}, {7,0,1,4,0,4}, {7,0,1,4,1,2}, {7,0,1,4,2,0}, {7,0,1,5,0,0},
  {7,0,2,0,2,8}, {7,0,2,0,3,6}, {7,0,2,0,4,4}, {7,0,2,0,5,2}, {7,0,2,0,6,0},
  {7,0,2,1,0,8}, {7,0,2,1,1,6}, {7,0,2,1,2,4}, {7,0,2,1,3,2}, {7,0,2,1,4,0},
  {7,0,2,2,0,4}, {7,0,2,2,1,2}, {7,0,2,2,2,0}, {7,0,2,3,0,0}, {7,0,3,0,0,4},
  {7,0,3,0,1,2}, {7,0,3,0,2,0}, {7,0,3,1,0,0}, {7,1,0,0,2,8}, {7,1,0,0,3,6},
  {7,1,0,0,4,4}, {7,1,0,0,5,2}, {7,1,0,0,6,0}, {7,1,0,1,0,8}, {7,1,0,1,1,6},
  {7,1,0,1,2,4}, {7,1,0,1,3,2}, {7,1,0,1,4,0}, {7,1,0,2,0,4}, {7,1,0,2,1,2},
  {7,1,0,2,2,0}, {7,1,0,3,0,0}, {7,1,1,0,0,4}, {7,1,1,0,1,2}, {7,1,1,0,2,0},
  {7,1,1,1,0,0}]
* Count = 2081

 [Order 9; Base 3]
If [27*An+9*Bn+3*Cn+Dn==360] then [{An,Bn,Cn,Dn} =
  { 5,17,18,18}, { 5,18,15,18}, { 5,18,16,15}, { 5,18,17,12}, { 5,18,18, 9},
  { 6,14,18,18}, { 6,15,15,18}, { 6,15,16,15}, { 6,15,17,12}, { 6,15,18, 9},
  { 6,16,12,18}, { 6,16,13,15}, { 6,16,14,12}, { 6,16,15, 9}, { 6,16,16, 6},
  { 6,16,17, 3}, { 6,16,18, 0}, { 6,17, 9,18}, { 6,17,10,15}, { 6,17,11,12},
  { 6,17,12, 9}, { 6,17,13, 6}, { 6,17,14, 3}, { 6,17,15, 0}, { 6,18, 6,18},
  { 6,18, 7,15}, { 6,18, 8,12}, { 6,18, 9, 9}, { 6,18,10, 6}, { 6,18,11, 3},
  { 6,18,12, 0}, { 7,11,18,18}, { 7,12,15,18}, { 7,12,16,15}, { 7,12,17,12},
  { 7,12,18, 9}, { 7,13,12,18}, { 7,13,13,15}, { 7,13,14,12}, { 7,13,15, 9},
  { 7,13,16, 6}, { 7,13,17, 3}, { 7,13,18, 0}, { 7,14, 9,18}, { 7,14,10,15},
  { 7,14,11,12}, { 7,14,12, 9}, { 7,14,13, 6}, { 7,14,14, 3}, { 7,14,15, 0},
  { 7,15, 6,18}, { 7,15, 7,15}, { 7,15, 8,12}, { 7,15, 9, 9}, { 7,15,10, 6},
  { 7,15,11, 3}, { 7,15,12, 0}, { 7,16, 3,18}, { 7,16, 4,15}, { 7,16, 5,12},
  { 7,16, 6, 9}, { 7,16, 7, 6}, { 7,16, 8, 3}, { 7,16, 9, 0}, { 7,17, 0,18},
  { 7,17, 1,15}, { 7,17, 2,12}, { 7,17, 3, 9}, { 7,17, 4, 6}, { 7,17, 5, 3},
  { 7,17, 6, 0}, { 7,18, 0, 9}, { 7,18, 1, 6}, { 7,18, 2, 3}, { 7,18, 3, 0},
  { 8, 8,18,18}, { 8, 9,15,18}, { 8, 9,16,15}, { 8, 9,17,12}, { 8, 9,18, 9},
  { 8,10,12,18}, { 8,10,13,15}, { 8,10,14,12}, { 8,10,15, 9}, { 8,10,16, 6},
  { 8,10,17, 3}, { 8,10,18, 0}, { 8,11, 9,18}, { 8,11,10,15}, { 8,11,11,12},
  { 8,11,12, 9}, { 8,11,13, 6}, { 8,11,14, 3}, { 8,11,15, 0}, { 8,12, 6,18},
  { 8,12, 7,15}, { 8,12, 8,12}, { 8,12, 9, 9}, { 8,12,10, 6}, { 8,12,11, 3},
  { 8,12,12, 0}, { 8,13, 3,18}, { 8,13, 4,15}, { 8,13, 5,12}, { 8,13, 6, 9},
  { 8,13, 7, 6}, { 8,13, 8, 3}, { 8,13, 9, 0}, { 8,14, 0,18}, { 8,14, 1,15},
  { 8,14, 2,12}, { 8,14, 3, 9}, { 8,14, 4, 6}, { 8,14, 5, 3}, { 8,14, 6, 0},
  { 8,15, 0, 9}, { 8,15, 1, 6}, { 8,15, 2, 3}, { 8,15, 3, 0}, { 8,16, 0, 0},
  { 9, 5,18,18}, { 9, 6,15,18}, { 9, 6,16,15}, { 9, 6,17,12}, { 9, 6,18, 9},
  { 9, 7,12,18}, { 9, 7,13,15}, { 9, 7,14,12}, { 9, 7,15, 9}, { 9, 7,16, 6},
  { 9, 7,17, 3}, { 9, 7,18, 0}, { 9, 8, 9,18}, { 9, 8,10,15}, { 9, 8,11,12},
```

```
  { 9, 8,12, 9}, { 9, 8,13, 6}, { 9, 8,14, 3}, { 9, 8,15, 0}, { 9, 9, 6,18},
  { 9, 9, 7,15}, { 9, 9, 8,12}, { 9, 9, 9, 9}, { 9, 9,10, 6}, { 9, 9,11, 3},
  { 9, 9,12, 0}, { 9,10, 3,18}, { 9,10, 4,15}, { 9,10, 5,12}, { 9,10, 6, 9},
  { 9,10, 7, 6}, { 9,10, 8, 3}, { 9,10, 9, 0}, { 9,11, 0,18}, { 9,11, 1,15},
  { 9,11, 2,12}, { 9,11, 3, 9}, { 9,11, 4, 6}, { 9,11, 5, 3}, { 9,11, 6, 0},
  { 9,12, 0, 9}, { 9,12, 1, 6}, { 9,12, 2, 3}, { 9,12, 3, 0}, { 9,13, 0, 0},
  {10, 2,18,18}, {10, 3,15,18}, {10, 3,16,15}, {10, 3,17,12}, {10, 3,18, 9},
  {10, 4,12,18}, {10, 4,13,15}, {10, 4,14,12}, {10, 4,15, 9}, {10, 4,16, 6},
  {10, 4,17, 3}, {10, 4,18, 0}, {10, 5, 9,18}, {10, 5,10,15}, {10, 5,11,12},
  {10, 5,12, 9}, {10, 5,13, 6}, {10, 5,14, 3}, {10, 5,15, 0}, {10, 6, 6,18},
  {10, 6, 7,15}, {10, 6, 8,12}, {10, 6, 9, 9}, {10, 6,10, 6}, {10, 6,11, 3},
  {10, 6,12, 0}, {10, 7, 3,18}, {10, 7, 4,15}, {10, 7, 5,12}, {10, 7, 6, 9},
  {10, 7, 7, 6}, {10, 7, 8, 3}, {10, 7, 9, 0}, {10, 8, 0,18}, {10, 8, 1,15},
  {10, 8, 2,12}, {10, 8, 3, 9}, {10, 8, 4, 6}, {10, 8, 5, 3}, {10, 8, 6, 0},
  {10, 9, 0, 9}, {10, 9, 1, 6}, {10, 9, 2, 3}, {10, 9, 3, 0}, {10,10, 0, 0},
  {11, 0,15,18}, {11, 0,16,15}, {11, 0,17,12}, {11, 0,18, 9}, {11, 1,12,18},
  {11, 1,13,15}, {11, 1,14,12}, {11, 1,15, 9}, {11, 1,16, 6}, {11, 1,17, 3},
  {11, 1,18, 0}, {11, 2, 9,18}, {11, 2,10,15}, {11, 2,11,12}, {11, 2,12, 9},
  {11, 2,13, 6}, {11, 2,14, 3}, {11, 2,15, 0}, {11, 3, 6,18}, {11, 3, 7,15},
  {11, 3, 8,12}, {11, 3, 9, 9}, {11, 3,10, 6}, {11, 3,11, 3}, {11, 3,12, 0},
  {11, 4, 3,18}, {11, 4, 4,15}, {11, 4, 5,12}, {11, 4, 6, 9}, {11, 4, 7, 6},
  {11, 4, 8, 3}, {11, 4, 9, 0}, {11, 5, 0,18}, {11, 5, 1,15}, {11, 5, 2,12},
  {11, 5, 3, 9}, {11, 5, 4, 6}, {11, 5, 5, 3}, {11, 5, 6, 0}, {11, 6, 0, 9},
  {11, 6, 1, 6}, {11, 6, 2, 3}, {11, 6, 3, 0}, {11, 7, 0, 0}, {12, 0, 6,18},
  {12, 0, 7,15}, {12, 0, 8,12}, {12, 0, 9, 9}, {12, 0,10, 6}, {12, 0,11, 3},
  {12, 0,12, 0}, {12, 1, 3,18}, {12, 1, 4,15}, {12, 1, 5,12}, {12, 1, 6, 9},
  {12, 1, 7, 6}, {12, 1, 8, 3}, {12, 1, 9, 0}, {12, 2, 0,18}, {12, 2, 1,15},
  {12, 2, 2,12}, {12, 2, 3, 9}, {12, 2, 4, 6}, {12, 2, 5, 3}, {12, 2, 6, 0},
  {12, 3, 0, 9}, {12, 3, 1, 6}, {12, 3, 2, 3}, {12, 3, 3, 0}, {12, 4, 0, 0},
  {13, 0, 0, 9}, {13, 0, 1, 6}, {13, 0, 2, 3}, {13, 0, 3, 0}, {13, 1, 0, 0}]
* Count = 285
```

Why! There are so many value patterns for order 8 as 2081. The only pattern {4, 4, 4, 4, 4, 4} could make the Complete Euler Squares. I think it may explain well about the fact Non-C.E.S. should exist far more than C.E.S. and the latter type could rarely be found.

It is practically impossible for you to compose all solutions of Non-C.E.S. of order 8. It may take a long, long time to count up to the last, even when you can use your super personal computer. It is also impossible for you to save all the solution data to outer memory devices. You cannot classify all into any small groups, and cannot study them analytically one by one, either.

It is no use of building Non-Euler Squares, I suppose. I recommend you not to try to count them all. You should rather compose only Complete Euler Squares and study them well by your own style.

(First written on November 15, 2003 by Kanji Setsuda with MWCW_for_Mac;
Retyped on March 26, 2007 by Kanji Setsuda with MacOSX and Xcode2.2.1)

E-Mail Address <jag12001@nifty.com>