

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

### Chapter 7: New Method of Composing High-Dimensional Extra-Cubic Objects and their Developed Forms: **Kanji Setsuda**

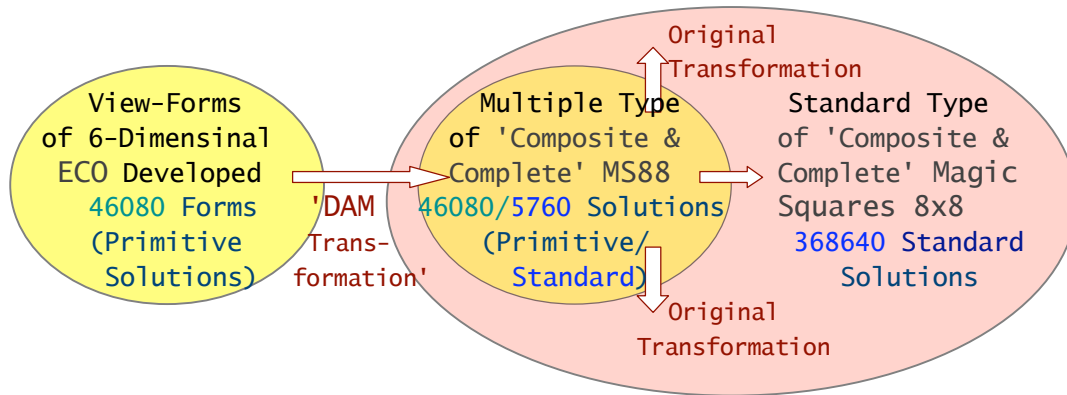
#### Section 6-3: New Method of Making Standard 'C&C' Magic Squares 8<sup>2</sup> with All View-Forms of ECO<sup>2</sup> and Additional Transformation System

##### #1. Our New Object in this Section

Let's compose another common type of magic things, some typical magic squares of order 8 here, using all view-forms of 6 dimensional Extra-Cubic Objects developed, the 'DAM Transformation' and our new additional transformation system.

I would like to report about this newest method of mine and some actual result of its application right here in this section.

Let's make the 368640 'Composite & Complete' Magic Squares of Order 8, shall we?



##### #2. What is the Difference between 'Multiple' C&C MS88 and 'Standard' One?

What is the 'Standard Type' of C&C Magic Squares of Order 8, then? It is known it has the larger set of 368640 standard solutions.

What do they look like? Which part is different from the 'Multiple 4x4' type?

This is the problem of definitions. Let me explain about Multiple type at first.

```
/** Multiple 4x4 Type of 'Composite & Complete' Magic Squares 8x8: **/
/* Basic Form *                               * Solution Example */
```

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

```
** Basic Conditions for Multiple 4x4 Type of PMS88: S=130; LS=260 **
n1+n2+n3+n4=S      ...rw1; | n1+n9+n17+n25=S      ...c11;
n5+n6+n7+n8=S      ...rw2; | n33+n41+n49+n57=S      ...c12;
```



```

/** Standard Type of 'Composite & Complete' Magic Squares 8x8: **/
/* Basic Form *                               * Solution Example */
61 62 63 64 57 58 59 60 61 62 63 64 57 58 59 60
 5  6  7  8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1  2  3  4
13 14 15 16 | 9 |10|11|12|13|14|15|16 | 9 10 11 12
21 22 23 24 |17|18|19|20|21|22|23|24 |17 18 19 20
29 30 31 32 |25|26|27|28|29|30|31|32 |25 26 27 28
37 38 39 40 |33|34|35|36|37|38|39|40 |33 34 35 36
45 46 47 48 |41|42|43|44|45|46|47|48 |41 42 43 44
53 54 55 56 |49|50|51|52|53|54|55|56 |49 50 51 52
61 62 63 64 |57|58|59|60|61|62|63|64 |57 58 59 60
 5  6  7  8 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 1  2  3  4

```

```

** Basic Conditions for Standard Type of PMS88: LS=260 **
n1+n2+n3+n4+n5+n6+n7+n8=LS      ...rw1;| n1+n9+n17+n25+n33+n41+n49+n57=LS ...c1;
n1+n10+n19+n28+n37+n46+n55+n64=LS ...pd1;| n8+n15+n22+n29+n36+n43+n50+n57=LS ...pb8;

```

There are no other differences between the two types. Both 'Composite Conditions' and 'Complete Conditions' should be the same with each other.

```

** Sample List: Standard Type of 'Composite & Complete' MS88 **

```

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

Those two types look very similar to each other. In fact, all of the Multiple type are included in the Standard one, as a special type of 'C&C' MS88. The solution set of 5760 Multiple type certainly makes the sub-set of 368640 Standard one.

If you want to have more information about them, go to Part 3 and read the old articles of mine in Section 6 in this site, please.

### #3. How to Compose those 'Standard' C&C Magic Squares of Order 8

I want to make them from the solution set of 'Multiple 4x4 type' of C&C MS88 and transform each into Standard type of C&C MS88 directly.

We already know how to compose the 5760 Multiple 4x4 type of C&C MS88 by our newest method using all possible view-forms of 6-dimensional ECO developed and 'Do-it-After-the-Model Transformation', just in the way I explained about before.

I want to know how to make these solutions into so many counts of Standard type as 368640, say, 64 times as many as 5760 of Multiple type.

The problem is how to find such a good transformation system to get the result.

I didn't want to have such the independent 5760 ways of transformation for each solution, but wanted to have a single system to deal with them totally. I tried and tried to find such a magic for a long time, but always failed in vain. It seemed any single transformation system was impossible for us to reach the goal.

I decided to change my idea. How about taking the combination of several possible ways to get the result? How about making a little too many answers at first to check and choose the true ones out of them afterward? How about trying to find some realistic, effective tools instead of looking for the idealistic, only one method.

I examined my new idea by making various experiments as hard as possible.

I could have finally found how to combine several possible transformation methods and have successfully got the complete solution set of 368640 Standard 'C&C' MS88.

I have really got the two effective sets of transformation methods, but let me introduce you one of them as a sample here in this section.

#### #4. About the Transformation using Unique Pan-magic Line Rotation

First of all, let me explain about the old transformation method that repeats unique Line Rotations for any pan-magic squares. It is regarded as the most essential method of transformation. It can produce 8x8 different forms from the original one.

The next two sets of diagrams express the concept of Pan-magic Line Rotations.

**/\*\* Conceptual Diagrams for Basic Transformation: Pan-magic Line Rotations \*\*/**

**\* Rotation 1: Bottom to Top \***

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

**\* Rotation 2: Left to Right \***

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

If you combine these two types of Line Rotations one by one, you should be able to produce the new set of 8x8 solutions from the original one.

Any of the solutions produced by this method still remain a Complete MS88. They should break neither Composite Conditions nor Complete Conditions.

It is certified by the very conditions for the constant sum 260 that every 8 numbers on all the 16 pandiagonals must equally add up to.

But I have to say it is not perfect. We cannot use this for our purpose as a single method of transformation to get to the goal.

Watch the next diagrams. They appear when you use this method every four times. As far as the origin is of multiple type, each of the other three remain also multiple.

**\*\* Conceptual Diagrams for the 4 Derivative Forms of 'Multiple' C&C MS88 \*\***

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

Four multiple types of product should be too many for us, because we wanted to produce 63 non-multiple types from the multiple origin by our new transformation.

**#5. About Our Newest Transformation System**

I had to find any other additional types of transformation method.

Let me introduce you such a set of three types as listed below.

**/\*\* Conceptual Diagrams for the Transformation from 1 'Multiple' into 4 'Standard' \*\*/**

* List 1 *								000/								010/								020/								030/							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	39	38	4	5	35	34	8	1	39	38	4	5	35	34	8								
9	10	11	12	13	14	15	16	53	54	55	56	49	50	51	52	9	47	46	12	13	43	42	16	53	19	18	56	49	23	22	52								
17	18	19	20	21	22	23	24	45	46	47	48	41	42	43	44	17	55	54	20	21	51	50	24	45	11	10	48	41	15	14	44								
25	26	27	28	29	30	31	32	25	26	27	28	29	30	31	32	25	63	62	28	29	59	58	32	25	63	62	28	29	59	58	32								
33	34	35	36	37	38	39	40	33	34	35	36	37	38	39	40	33	7	6	36	37	3	2	40	33	7	6	36	37	3	2	40								
41	42	43	44	45	46	47	48	21	22	23	24	17	18	19	20	41	15	14	44	45	11	10	48	21	51	50	24	17	55	54	20								
49	50	51	52	53	54	55	56	13	14	15	16	9	10	11	12	49	23	22	52	53	19	18	56	13	43	42	16	9	47	46	12								
57	58	59	60	61	62	63	64	57	58	59	60	61	62	63	64	57	31	30	60	61	27	26	64	57	31	30	60	61	27	26	64								
* List 1R *								1/								65/								129/								193/							
1	63	4	62	8	58	5	59	1	63	4	62	8	58	5	59	1	61	2	62	8	60	7	59	1	61	2	62	8	60	7	59								
56	10	53	11	49	15	52	14	40	26	37	27	33	31	36	30	56	12	55	11	49	13	50	14	40	28	39	27	33	29	34	30								
25	39	28	38	32	34	29	35	9	55	12	54	16	50	13	51	25	37	26	38	32	36	31	35	9	53	10	54	16	52	15	51								
48	18	45	19	41	23	44	22	48	18	45	19	41	23	44	22	48	20	47	19	41	21	42	22	48	20	47	19	41	21	42	22								
57	7	60	6	64	2	61	3	57	7	60	6	64	2	61	3	57	5	58	6	64	4	63	3	57	5	58	6	64	4	63	3								
16	50	13	51	9	55	12	54	32	34	29	35	25	39	28	38	16	52	15	51	9	53	10	54	32	36	31	35	25	37	26	38								
33	31	36	30	40	26	37	27	49	15	52	14	56	10	53	11	33	29	34	30	40	28	39	27	49	13	50	14	56	12	55	11								
24	42	21	43	17	47	20	46	24	42	21	43	17	47	20	46	24	44	23	43	17	45	18	46	24	44	23	43	17	45	18	46								

Diagram 000/ illustrates the Basic Form for the concept of original solution, and those 010/~030/ express how to make their transformations individually.

Each of 010/ and 020/ expresses the same kind of transformation, only different in directions. Both of them keep half unchanged and change the rest half of the numbers. Watch them and know how 32 don't move and how the rest 32 move.

The last 030/ is the combined transformation of 010/ and 020/, as you see.

Four diagrams of List 1R above shows the real solutions of our object C&C MS88 corresponding to those transformations.

How did I find these transformation methods? Let me illustrate as below.

I learned them from the list of real object solutions, comparing some similar pairs of them, and took their transformation rules between them.

```

/** Compare the Similar Pair Solutions and Take out their Transformation Rules */
** Transformation Type19 **
      1/B              1/              19/              19/T
  1  2  3  4  5  6  7  8   1 63  4 62  8 58  5 59   1 63  4 62  8 58  5 59   1  2  3  4  5  6  7  8
  9 10 11 12 13 14 15 16   56 10 53 11 49 15 52 14   40 26 37 27 33 31 36 30   53 54 55 56 49 50 51 52
 17 18 19 20 21 22 23 24   25 39 28 38 32 34 29 35     9 55 12 54 16 50 13 51   45 46 47 48 41 42 43 44
 25 26 27 28 29 30 31 32   48 18 45 19 41 23 44 22   48 18 45 19 41 23 44 22   25 26 27 28 29 30 31 32
 33 34 35 36 37 38 39 40   57  7 60  6 64  2 61  3   57  7 60  6 64  2 61  3   33 34 35 36 37 38 39 40
 41 42 43 44 45 46 47 48   16 50 13 51  9 55 12 54   32 34 29 35 25 39 28 38   21 22 23 24 17 18 19 20
 49 50 51 52 53 54 55 56   33 31 36 30 40 26 37 27   49 15 52 14 56 10 53 11   13 14 15 16  9 10 11 12
 57 58 59 60 61 62 63 64   24 42 21 43 17 47 20 46   24 42 21 43 17 47 20 46   57 58 59 60 61 62 63 64
/**/
void trnsf19(){
  mm[1]=nm[1];   mm[2]=nm[2];   mm[3]=nm[3];   mm[4]=nm[4];
  mm[5]=nm[5];   mm[6]=nm[6];   mm[7]=nm[7];   mm[8]=nm[8];
  mm[9]=nm[53];  mm[10]=nm[54]; mm[11]=nm[55]; mm[12]=nm[56];
  mm[13]=nm[49]; mm[14]=nm[50]; mm[15]=nm[51]; mm[16]=nm[52];
  mm[17]=nm[45]; mm[18]=nm[46]; mm[19]=nm[47]; mm[20]=nm[48];
  mm[21]=nm[41]; mm[22]=nm[42]; mm[23]=nm[43]; mm[24]=nm[44];
  mm[25]=nm[25]; mm[26]=nm[26]; mm[27]=nm[27]; mm[28]=nm[28];
  mm[29]=nm[29]; mm[30]=nm[30]; mm[31]=nm[31]; mm[32]=nm[32];
  mm[33]=nm[33]; mm[34]=nm[34]; mm[35]=nm[35]; mm[36]=nm[36];
  mm[37]=nm[37]; mm[38]=nm[38]; mm[39]=nm[39]; mm[40]=nm[40];
  mm[41]=nm[21]; mm[42]=nm[22]; mm[43]=nm[23]; mm[44]=nm[24];
  mm[45]=nm[17]; mm[46]=nm[18]; mm[47]=nm[19]; mm[48]=nm[20];
  mm[49]=nm[13]; mm[50]=nm[14]; mm[51]=nm[15]; mm[52]=nm[16];
  mm[53]=nm[9];  mm[54]=nm[10]; mm[55]=nm[11]; mm[56]=nm[12];
  mm[57]=nm[57]; mm[58]=nm[58]; mm[59]=nm[59]; mm[60]=nm[60];
  mm[61]=nm[61]; mm[62]=nm[62]; mm[63]=nm[63]; mm[64]=nm[64];
}
/**/

```

You have to dictate the same style of transformation rules for the other two.

It is important then for us to make the set of **four** types including the original one just like the List 1 in Page 5.

Our Standard type of C&C MS88 contains the Multiple 4x4 type as the sub-set of itself, so that we must not remove any one of multiple type from our object goal.

We use these transformation rules combining with the Pan-magic Line Rotation.

## #6. Job List: What are we going to do step by step?

(1) Make the developed diagrams of 6-dimensional Extra-Cubic Object of Order 2 as many as possible, just in the way as explained before in the previous section.

We have to have so many diagrams as 46080 for the 'primitive solutions'.

We now regard these diagrams as 'Prototype Squares of Order 8', and are going to apply 'DAM Transformation' to each Prototype Square.

The 'Model Solution' must be prepared in advance by selecting out of the object.

The 'Do-it-After-the-Model Transformation' must also be defined as follows.

```

/* Model Solution and DAM Transformation */
** Prototype Square 0/      The Model Selected 0/ **
  1  2  3  4  5  6  7  8   1 63  4 62  8 58  5 59
  9 10 11 12 13 14 15 16   56 10 53 11 49 15 52 14
 17 18 19 20 21 22 23 24   25 39 28 38 32 34 29 35
 25 26 27 28 29 30 31 32   48 18 45 19 41 23 44 22
 33 34 35 36 37 38 39 40   57  7 60  6 64  2 61  3
 41 42 43 44 45 46 47 48   16 50 13 51  9 55 12 54
 49 50 51 52 53 54 55 56   33 31 36 30 40 26 37 27
 57 58 59 60 61 62 63 64   24 42 21 43 17 47 20 46
/**/
/* Dictated Program of DAM Transformation */

```

```

void trnsfdam(long x){
long int m,mc;
short n,sm[65];
for(m=0;m<x;m++){
sm[1]=snm[m][1]; sm[2]=snm[m][63]; sm[3]=snm[m][4]; sm[4]=snm[m][62];
sm[5]=snm[m][8]; sm[6]=snm[m][58]; sm[7]=snm[m][5]; sm[8]=snm[m][59];
sm[9]=snm[m][56]; sm[10]=snm[m][10]; sm[11]=snm[m][53]; sm[12]=snm[m][11];
sm[13]=snm[m][49]; sm[14]=snm[m][15]; sm[15]=snm[m][52]; sm[16]=snm[m][14];
sm[17]=snm[m][25]; sm[18]=snm[m][39]; sm[19]=snm[m][28]; sm[20]=snm[m][38];
sm[21]=snm[m][32]; sm[22]=snm[m][34]; sm[23]=snm[m][29]; sm[24]=snm[m][35];
sm[25]=snm[m][48]; sm[26]=snm[m][18]; sm[27]=snm[m][45]; sm[28]=snm[m][19];
sm[29]=snm[m][41]; sm[30]=snm[m][23]; sm[31]=snm[m][44]; sm[32]=snm[m][22];
sm[33]=snm[m][57]; sm[34]=snm[m][7]; sm[35]=snm[m][60]; sm[36]=snm[m][6];
sm[37]=snm[m][64]; sm[38]=snm[m][2]; sm[39]=snm[m][61]; sm[40]=snm[m][3];
sm[41]=snm[m][16]; sm[42]=snm[m][50]; sm[43]=snm[m][13]; sm[44]=snm[m][51];
sm[45]=snm[m][9]; sm[46]=snm[m][55]; sm[47]=snm[m][12]; sm[48]=snm[m][54];
sm[49]=snm[m][33]; sm[50]=snm[m][31]; sm[51]=snm[m][36]; sm[52]=snm[m][30];
sm[53]=snm[m][40]; sm[54]=snm[m][26]; sm[55]=snm[m][37]; sm[56]=snm[m][27];
sm[57]=snm[m][24]; sm[58]=snm[m][42]; sm[59]=snm[m][21]; sm[60]=snm[m][43];
sm[61]=snm[m][17]; sm[62]=snm[m][47]; sm[63]=snm[m][20]; sm[64]=snm[m][46];
for(n=1;n<65;n++){tnm[m][n]=sm[n];}
}
}
/**/

```

(2) Make the 46080 primitive solutions of Multiple 'C&C' MS88 by applying 'DAMT' to the 46080 Prototype Squares. Save all the results to our table to re-use afterward.

We now prefer not to use any 'List-forming Inequality Conditions', and not to select the 5760 'standard solutions' by them here.

I fear those inequality conditions may sometimes conflict with any transformation process and make serious troubles.

I like to show you the process on the way in the list below by some sample pairs of Prototype Squares and the corresponding Multiple C&C MS88 transformed.

These 20 prototypes are so rare and beautiful that they are sometimes called 'Principal Squares' for the 'Fundamental Solutions' of C&C MS88.

**\*\* List of Prototype Squares 8x8 and Multiple 'C&C' MS88 Transformed \*\***

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

	52/P	T/		61/P	T/																		
1	3	5	7	17	19	21	23	1	2	9	10	17	18	25	26	1	63	10	56	26	40	17	47
2	4	6	8	18	20	22	24	63	4	57	6	41	22	47	20	62	4	53	11	37	27	46	20
9	11	13	15	25	27	29	31	10	53	16	51	32	35	26	37	7	57	16	50	32	34	23	41
10	12	14	16	26	28	30	32	56	11	50	13	34	29	40	27	60	6	51	13	35	29	44	22
33	35	37	39	49	51	53	55	42	21	48	19	64	3	58	5	39	25	48	18	64	2	55	9
34	36	38	40	50	52	54	56	24	43	18	45	2	61	8	59	28	38	19	45	3	61	12	54
41	43	45	47	57	59	61	63	33	30	39	28	55	12	49	14	33	31	42	24	58	8	49	15
42	44	46	48	58	60	62	64	31	36	25	38	9	54	15	52	30	36	21	43	5	59	14	52
	62/P	T/		65/P	T/																		
1	3	9	11	17	19	25	27	1	62	11	56	27	40	17	46	1	60	13	56	29	40	17	44
2	4	10	12	18	20	26	28	63	4	53	10	37	26	47	20	63	6	51	10	35	26	47	22
5	7	13	15	21	23	29	31	6	57	16	51	32	35	22	41	4	57	16	53	32	37	20	41
6	8	14	16	22	24	30	32	60	7	50	13	34	29	44	23	62	7	50	11	34	27	46	23
33	35	41	43	49	51	57	59	38	25	48	19	64	3	54	9	36	25	48	21	64	5	52	9
34	36	42	44	50	52	58	60	28	39	18	45	2	61	12	55	30	39	18	43	2	59	14	55
37	39	45	47	53	55	61	63	33	30	43	24	59	8	49	14	33	28	45	24	61	8	49	12
38	40	46	48	54	56	62	64	31	36	21	42	5	58	15	52	31	38	19	42	3	58	15	54
	361/P	T/		363/P	T/																		
1	2	3	4	33	34	35	36	1	63	4	62	36	30	33	31	1	63	6	60	38	28	33	31
5	6	7	8	37	38	39	40	60	6	57	7	25	39	28	38	62	4	57	7	25	39	30	36
9	10	11	12	41	42	43	44	13	51	16	50	48	18	45	19	11	53	16	50	48	18	43	21
13	14	15	16	45	46	47	48	56	10	53	11	21	43	24	42	56	10	51	13	19	45	24	42
17	18	19	20	49	50	51	52	29	35	32	34	64	2	61	3	27	37	32	34	64	2	59	5
21	22	23	24	53	54	55	56	40	26	37	27	5	59	8	58	40	26	35	29	3	61	8	58
25	26	27	28	57	58	59	60	17	47	20	46	52	14	49	15	17	47	22	44	54	12	49	15
29	30	31	32	61	62	63	64	44	22	41	23	9	55	12	54	46	20	41	23	9	55	14	52
	364/P	T/		373/P	T/																		
1	3	5	7	33	35	37	39	1	62	7	60	39	28	33	30	1	63	10	56	42	24	33	31
2	4	6	8	34	36	38	40	63	4	57	6	25	38	31	36	62	4	53	11	21	43	30	36
9	11	13	15	41	43	45	47	10	53	16	51	48	19	42	21	7	57	16	50	48	18	39	25
10	12	14	16	42	44	46	48	56	11	50	13	18	45	24	43	60	6	51	13	19	45	28	38
17	19	21	23	49	51	53	55	26	37	32	35	64	3	58	5	23	41	32	34	64	2	55	9
18	20	22	24	50	52	54	56	40	27	34	29	2	61	8	59	44	22	35	29	3	61	12	54
25	27	29	31	57	59	61	63	17	46	23	44	55	12	49	14	17	47	26	40	58	8	49	15
26	28	30	32	58	60	62	64	47	20	41	22	9	54	15	52	46	20	37	27	5	59	14	52
	374/P	T/		377/P	T/																		
1	3	9	11	33	35	41	43	1	62	11	56	43	24	33	30	1	60	13	56	45	24	33	28
2	4	10	12	34	36	42	44	63	4	53	10	21	42	31	36	63	6	51	10	19	42	31	38
5	7	13	15	37	39	45	47	6	57	16	51	48	19	38	25	4	57	16	53	48	21	36	25
6	8	14	16	38	40	46	48	60	7	50	13	18	45	28	39	62	7	50	11	18	43	30	39
17	19	25	27	49	51	57	59	22	41	32	35	64	3	54	9	20	41	32	37	64	5	52	9
18	20	26	28	50	52	58	60	44	23	34	29	2	61	12	55	46	23	34	27	2	59	14	55
21	23	29	31	53	55	61	63	17	46	27	40	59	8	49	14	17	44	29	40	61	8	49	12
22	24	30	32	54	56	62	64	47	20	37	26	5	58	15	52	47	22	35	26	3	58	15	54
	433/P	T/		434/P	T/																		
1	2	17	18	33	34	49	50	1	63	18	48	50	16	33	31	1	62	19	48	51	16	33	30
3	4	19	20	35	36	51	52	62	4	45	19	13	51	30	36	63	4	45	18	13	50	31	36
5	6	21	22	37	38	53	54	7	57	24	42	56	10	39	25	6	57	24	43	56	11	38	25
7	8	23	24	39	40	55	56	60	6	43	21	11	53	28	38	60	7	42	21	10	53	28	39
9	10	25	26	41	42	57	58	15	49	32	34	64	2	47	17	14	49	32	35	64	3	46	17
11	12	27	28	43	44	59	60	52	14	35	29	3	61	20	46	10	12	26	28	42	44	58	60
13	14	29	30	45	46	61	62	9	55	26	40	58	8	41	23	13	15	29	31	45	47	61	63
15	16	31	32	47	48	63	64	54	12	37	27	5	59	22	44	14	16	30	32	46	48	62	64
	437/P	T/		451/P	T/																		
1	5	17	21	33	37	49	53	1	60	21	48	53	16	33	28	1	9	17	25	33	41	49	57
2	6	18	22	34	38	50	54	63	6	43	18	11	50	31	38	2	10	18	26	34	42	50	58
3	7	19	23	35	39	51	55	4	57	24	45	56	13	36	25	3	11	19	27	35	43	51	59
4	8	20	24	36	40	52	56	62	7	42	19	10	51	30	39	4	12	20	28	36	44	52	60
9	13	25	29	41	45	57	61	12	49	32	37	64	5	44	17	5	13	21	29	37	45	53	61
10	14	26	30	42	46	58	62	54	15	34	27	2	59	22	47	6	14	22	30	38	46	54	62
11	15	27	31	43	47	59	63	9	52	29	40	61	8	41	20	7	15	23	31	39	47	55	63
12	16	28	32	44	48	60	64	55	14	35	26	3	58	23	46	8	16	24	32	40	48	56	64

Do you notice 451/ is the reflected pattern of 1/, 437/ is the reflected one of 7/, 434/ is also the reflected one of 9/, and so on? Among the set of primitive solutions there are such reflected patterns or rotated ones, so many as 8 derivatives for each.

We are going to do nothing about it but save all primitive answers to our table.

Let it be the last job for us to select the standard solutions, to be done in the end.

(3) It is the time now we use our newest transformation system combining two groups of methods, and make those Multiple C&C MS88 transformed into our last goal Standard C&C MS88. We will have finally got the smart list of 368640 objects selected under the list-forming inequality conditions.

## #7. Program List

Let me show you the list of compact program I wrote for this experiment.

The first half of this program is almost the same as the one I put in the previous section. I really added the last half to it for my new transformation process.

```

/** Compose 'Multiple' type of 'Composite & Complete' Magic Squares of Order 8 */
/** by New Method Using All Possible View-Forms of Developed Extra-Cubic Objects */
/** of Order 2^6 and 'Do-it-After-the-Model' Transformation, and then Transform */
/** 'Multiple' type to get into 368640 'Standard' type of 'C&C' MS88 in the end */
/** 'NewMS8CCTT.c' dictated by Kanji Setsuda */
/** Working with MacOSX10.4-10.5 & Xcode2-3 */
/** on Oct.2, '05; Optimized on Jan.17, '09 */
/**/
#include <stdio.h>
/**/
/* Global Variables */
long int cntr, cnt2;
short nm[65], mm[65];
short vc, pc;
short tv[65][6];
short td[721][6];
short tnm[46081][65], snm[46081][65];
short strn[368641][65];
short trn[257][65];
short cn1[65];
long tcn1[33]={0,
  23040, 46080, 68736, 91392, 113088, 134784, 156096, 177408,
  195648, 213888, 231744, 249600, 266496, 283392, 299904, 316416,
  322944, 329472, 335616, 341760, 346944, 352128, 356928, 361728,
  363456, 365184, 366528, 367872, 368256, 368640, 368640};
/**/
/* Sub-Routiens */
void mkfv26(void), prmt26(void);
void d26(short x, short y);
void trnsfdamt(long x);
/**/
void regsol(short x);
void prsol(long x, short y);
void prcn1(long x, short y);
void srt82(void), exc82(long x);
/**/
void trnsfmtn(void);
void trnsf100(void), trnsf200(void), trnsf300(void);
void trnsfpd(short x);
/**/
/* Main Program */
int main(){
  long int m,m1;
  short n,r,m2;
  printf("\n*** Compose 'Multiple' type of 'Composite & Complete' Magic Squares 8x8 ***\n");
  printf("*** by New Method Using All Possible View-Forms of Developed Extra-Cubic ***\n");
  printf("*** Objects of Order 2^6 and 'Do-it-After-the-Model' Transformation ***\n");
  for(n=0;n<65;n++){cn1[n]=0;}
  vc=0; mkfv26();
  pc=0; prmt26();
  cntr=vc*pc;
  for(m=0;m<vc;m++){for(n=0;n<pc;n++){d26(m,n);}}
  trnsfdamt(cntr);

```

```

printf("\n** List of Solution Counts according to the Values of N1: **\n");
n=cntr/720; prcn1(cntr,n);
/**/
printf("\n** Transform 'Multiple' type into 'Standard' type of 'C&C' Magic Squares 8x8\n");
printf(" by the Pan-magic Line Rotation Plus Transformation System from 1 into 4x64 **\n");
printf(" ... Monitoring Total Solution Count in Every Step:\n");
cnt2=cntr/2; cntr=0;
for(n=0;n<65;n++){cn1[n]=0;}
for(m=1;m<368641;m++){strn[m][0]=0; strn[m][1]=0;}
for(m=0;m<cnt2;m++){r=tnm[m][1];
  if((0<r)&&(r<33)){
    for(n=1;n<65;n++){trn[0][n]=tnm[m][n];}
    trnsfmtn();
    regsol(256);}
  m1=m+1;
  if(m1%720==0){m2=m1/720;
    printf(" [%2d:%7d]",m2,cntr);
    if(m2%4==0){printf("\n");}
  }
}
printf("\n ... Sorting Solution Data for the smart List of Standard Set.\n");
printf(" You may have to wait for a long time, I am afraid.\n");
srt82();
printf("\n** Standard 'Composite & Complete' MS88 Transformed from 'Multiple' type **\n");
printf(" ** Abstract List of Standard Solutions Recomposed: **\n");
prsol(1,36); printf(" . . . .\n");
prsol(tc1[1]/4-3,12); printf(" . . . .\n");
prsol(tc1[1]-3,12); printf(" . . . .\n");
prsol(tc1[2]-3,8); printf(" . . . .\n");
prsol(tc1[3]-3,8); printf(" . . . .\n");
prsol(cntr-11,12);
printf(" [Count = %d]\n",cntr);
printf("\n** List of Solution Counts according to the Values of N1: **\n");
prcn1(cntr,32);
printf("\n OK!\n");
return 0;
}
/**/
/* Sub-Routines */
/* New Method: Sub-Procedures for Extra-Cubic Objects of Order 2^6 */
/**/
void mkfv26(){
short d0,d1,d2,d3,d4,d5;
for(d0=0;d0<2;d0++){
for(d1=0;d1<2;d1++){
for(d2=0;d2<2;d2++){
for(d3=0;d3<2;d3++){
for(d4=0;d4<2;d4++){
for(d5=0;d5<2;d5++){
tv[vc][0]=d0; tv[vc][1]=d1; tv[vc][2]=d2;
tv[vc][3]=d3; tv[vc][4]=d4; tv[vc][5]=d5;
vc++;
}}}}}}
}
/**/
void prmt26(void){
short d0,d1,d2,d3,d4,d5,n;
short uflg[6];
for(n=0;n<6;n++){uflg[n]=0;}
for(d0=0;d0<6;d0++){
uflg[d0]=1;
for(d1=0;d1<6;d1++){
if(uflg[d1]==0){uflg[d1]=1;
for(d2=0;d2<6;d2++){
if(uflg[d2]==0){uflg[d2]=1;
for(d3=0;d3<6;d3++){

```

```

        if(uflg[d3]==0){uflg[d3]=1;
        for(d4=0;d4<6;d4++){
            if(uflg[d4]==0){uflg[d4]=1;
            for(d5=0;d5<6;d5++){
                if(uflg[d5]==0){uflg[d5]=1;
                td[pc][0]=d0; td[pc][1]=d1; td[pc][2]=d2;
                td[pc][3]=d3; td[pc][4]=d4; td[pc][5]=d5;
                pc++;
                uflg[d5]=0;
            }}
            uflg[d4]=0;
        }}
        uflg[d3]=0;
    }}
    uflg[d2]=0;
}}
uflg[d1]=0;
}}
uflg[d0]=0;
}
}
/**/
void d26(short x, short y){
    short d0,d1,d2,d3,d4,d5;
    short t0,t1,t2,t3,t4,t5;
    short c;
    short s[6], cd[2][2];
    cd[0][0]=0; cd[0][1]=1; cd[1][0]=1; cd[1][1]=0;
    c=0;
    for(d0=0;d0<2;d0++){
        for(d1=0;d1<2;d1++){
            for(d2=0;d2<2;d2++){
                for(d3=0;d3<2;d3++){
                    for(d4=0;d4<2;d4++){
                        for(d5=0;d5<2;d5++){c++;
                            s[0]=cd[tv[x][0]][d0]; s[1]=cd[tv[x][1]][d1]; s[2]=cd[tv[x][2]][d2];
                            s[3]=cd[tv[x][3]][d3]; s[4]=cd[tv[x][4]][d4]; s[5]=cd[tv[x][5]][d5];
                            t0=td[y][0]; t1=td[y][1]; t2=td[y][2]; t3=td[y][3]; t4=td[y][4]; t5=td[y][5];
                            snm[x*pc+y][c]=(((s[t0]*2+s[t1])*2+s[t2])*2+s[t3])*2+s[t4])*2+s[t5]+1;
                        }}
                    }}
                }}
            }}
        }}
    }
}
/**/
void trnsfdamt(long x){
    long int m,mc;
    short p[65],q[65];
    short n,q1;
    for(m=0;m<x;m++){
        for(n=1;n<65;n++){p[n]=snm[m][n];}
        q[1]=p[1]; q[2]=p[63]; q[3]=p[4]; q[4]=p[62];
        q[5]=p[8]; q[6]=p[58]; q[7]=p[5]; q[8]=p[59];
        q[9]=p[56]; q[10]=p[10]; q[11]=p[53]; q[12]=p[11];
        q[13]=p[49]; q[14]=p[15]; q[15]=p[52]; q[16]=p[14];
        q[17]=p[25]; q[18]=p[39]; q[19]=p[28]; q[20]=p[38];
        q[21]=p[32]; q[22]=p[34]; q[23]=p[29]; q[24]=p[35];
        q[25]=p[48]; q[26]=p[18]; q[27]=p[45]; q[28]=p[19];
        q[29]=p[41]; q[30]=p[23]; q[31]=p[44]; q[32]=p[22];
        q[33]=p[57]; q[34]=p[7]; q[35]=p[60]; q[36]=p[6];
        q[37]=p[64]; q[38]=p[2]; q[39]=p[61]; q[40]=p[3];
        q[41]=p[16]; q[42]=p[50]; q[43]=p[13]; q[44]=p[51];
        q[45]=p[9]; q[46]=p[55]; q[47]=p[12]; q[48]=p[54];
        q[49]=p[33]; q[50]=p[31]; q[51]=p[36]; q[52]=p[30];
        q[53]=p[40]; q[54]=p[26]; q[55]=p[37]; q[56]=p[27];
        q[57]=p[24]; q[58]=p[42]; q[59]=p[21]; q[60]=p[43];
        q[61]=p[17]; q[62]=p[47]; q[63]=p[20]; q[64]=p[46];

```

```

    q1=q[1]; mc=(q1-1)*720+cn1[q1];
    for(n=1;n<65;n++){tnm[mc][n]=q[n];}
    tnm[mc][0]=m+1;
    cn1[q1]++;
}
}
/**/
/* Sort the Solution Data for the smart List of Standard Set */
void srt82(void){
    short f;
    long m,n,d1,d2,d3,d4;
    m=cntr;
    do{f=0; m--;
        for(n=0;n<m;n++){
            if(strn[n][1]>strn[n+1][1]){exc82(n); f=1;}
            d1=((strn[n][2]*65+strn[n][4])*65+strn[n][8])*65+strn[n][7];
            d2=((strn[n+1][2]*65+strn[n+1][4])*65+strn[n+1][8])*65+strn[n+1][7];
            if((strn[n][1]==strn[n+1][1])&&(d1<d2)){exc82(n); f=1;}
            d3=((strn[n][9]*65+strn[n][25])*65+strn[n][57])*65+strn[n][49];
            d4=((strn[n+1][9]*65+strn[n+1][25])*65+strn[n+1][57])*65+strn[n+1][49];
            if((strn[n][1]==strn[n+1][1])&&(d1==d2)&&(d3<d4)){exc82(n); f=1;}
        }
    }while(f>0);
}
/**/
/* Exchange solution pairs according to their own values */
void exc82(long x){
    short p,sv;
    for(p=0;p<65;p++){sv=strn[x][p]; strn[x][p]=strn[x+1][p]; strn[x+1][p]=sv;}
}
/**/
/* Register the composed Solution to the Table after checking if it is really a new one */
void regsol(short x){
    long int nn,tab,tae;
    short m,n,p,mtc;
    for(m=0;m<x;m++){
        if((trn[m][1]<trn[m][64])&&(trn[m][1]<trn[m][8])&&(trn[m][1]<trn[m][57])&&(trn[m][9]<trn[m][2])){
            mtc=0; tab=tcn1[trn[m][1]-1]; tae=tcn1[trn[m][1]];
            for(nn=tab;nn<tae;nn++){
                if(strn[nn][1]==0){break;}
                for(p=1;p<65;p++){if(trn[m][p]==strn[nn][p]){mtc++;}else{mtc=0; break;}}
                if(mtc==64){break;}
            }
            if(mtc<64){
                for(n=1;n<65;n++){strn[nn][n]=trn[m][n];}
                cn1[trn[m][1]]++;
                cntr++;
            }
        }
    }
}
}
/**/
/* Print Programs */
/* For Standard Solutions of 'Standard' type of 'Composite & Complete' MS88: */
void prsol(long x, short y){
    long int m;
    short l,l8,n,p;
    for(m=x-1;m<x+y-1;m=m+4){
        for(n=0;n<4;n++){printf("%25d/",m+n+1);}
        printf("\n");
        for(l=0;l<8;l++){l8=l*8;
            for(n=0;n<4;n++){
                printf(" ");
                for(p=1;p<9;p++){printf("%3d",strn[m+n][l8+p]);}
            }
        }
        printf("\n");
    }
}

```

```

    }
  }
}
/**/
/* Print Solution Counts according to the Values of N1: */
void prcn1(long x, short y){
  short n;
  for(n=1;n<=y;n++){
    printf(" (%2d:%5d)",n,cn1[n]);
    if(n%8==0){printf("\n");}
  }
  printf(" Sum = %d\n",x);
}
/**/
/** New Transformation System from 'Multiple' type into 'Standard' */
void trnsf100(void){
  short n;
  for(n=0;n<65;n++){nm[n]=trn[0][n];}; trnsfpd(0);
  trnsf100(); for(n=0;n<65;n++){trn[64][n]=mm[n];}; trnsfpd(64);
  trnsf200(); for(n=0;n<65;n++){trn[128][n]=mm[n];}; trnsfpd(128);
  trnsf300(); for(n=0;n<65;n++){trn[192][n]=mm[n];}; trnsfpd(192);
}
/**/
void trnsf100(void){
  mm[1]=nm[1];    mm[2]=nm[2];    mm[3]=nm[3];    mm[4]=nm[4];
  mm[5]=nm[5];    mm[6]=nm[6];    mm[7]=nm[7];    mm[8]=nm[8];
  mm[9]=nm[53];   mm[10]=nm[54];   mm[11]=nm[55];  mm[12]=nm[56];
  mm[13]=nm[49];  mm[14]=nm[50];  mm[15]=nm[51];  mm[16]=nm[52];
  mm[17]=nm[45];  mm[18]=nm[46];  mm[19]=nm[47];  mm[20]=nm[48];
  mm[21]=nm[41];  mm[22]=nm[42];  mm[23]=nm[43];  mm[24]=nm[44];
  mm[25]=nm[25];  mm[26]=nm[26];  mm[27]=nm[27];  mm[28]=nm[28];
  mm[29]=nm[29];  mm[30]=nm[30];  mm[31]=nm[31];  mm[32]=nm[32];
  mm[33]=nm[33];  mm[34]=nm[34];  mm[35]=nm[35];  mm[36]=nm[36];
  mm[37]=nm[37];  mm[38]=nm[38];  mm[39]=nm[39];  mm[40]=nm[40];
  mm[41]=nm[21];  mm[42]=nm[22];  mm[43]=nm[23];  mm[44]=nm[24];
  mm[45]=nm[17];  mm[46]=nm[18];  mm[47]=nm[19];  mm[48]=nm[20];
  mm[49]=nm[13];  mm[50]=nm[14];  mm[51]=nm[15];  mm[52]=nm[16];
  mm[53]=nm[9];   mm[54]=nm[10];  mm[55]=nm[11];  mm[56]=nm[12];
  mm[57]=nm[57];  mm[58]=nm[58];  mm[59]=nm[59];  mm[60]=nm[60];
  mm[61]=nm[61];  mm[62]=nm[62];  mm[63]=nm[63];  mm[64]=nm[64];
}
void trnsf200(void){
  mm[1]=nm[1];    mm[2]=nm[39];   mm[3]=nm[38];   mm[4]=nm[4];
  mm[5]=nm[5];    mm[6]=nm[35];   mm[7]=nm[34];   mm[8]=nm[8];
  mm[9]=nm[9];    mm[10]=nm[47];  mm[11]=nm[46];  mm[12]=nm[12];
  mm[13]=nm[13];  mm[14]=nm[43];  mm[15]=nm[42];  mm[16]=nm[16];
  mm[17]=nm[17];  mm[18]=nm[55];  mm[19]=nm[54];  mm[20]=nm[20];
  mm[21]=nm[21];  mm[22]=nm[51];  mm[23]=nm[50];  mm[24]=nm[24];
  mm[25]=nm[25];  mm[26]=nm[63];  mm[27]=nm[62];  mm[28]=nm[28];
  mm[29]=nm[29];  mm[30]=nm[59];  mm[31]=nm[58];  mm[32]=nm[32];
  mm[33]=nm[33];  mm[34]=nm[7];   mm[35]=nm[6];   mm[36]=nm[36];
  mm[37]=nm[37];  mm[38]=nm[3];   mm[39]=nm[2];   mm[40]=nm[40];
  mm[41]=nm[41];  mm[42]=nm[15];  mm[43]=nm[14];  mm[44]=nm[44];
  mm[45]=nm[45];  mm[46]=nm[11];  mm[47]=nm[10];  mm[48]=nm[48];
  mm[49]=nm[49];  mm[50]=nm[23];  mm[51]=nm[22];  mm[52]=nm[52];
  mm[53]=nm[53];  mm[54]=nm[19];  mm[55]=nm[18];  mm[56]=nm[56];
  mm[57]=nm[57];  mm[58]=nm[31];  mm[59]=nm[30];  mm[60]=nm[60];
  mm[61]=nm[61];  mm[62]=nm[27];  mm[63]=nm[26];  mm[64]=nm[64];
}
void trnsf300(void){
  mm[1]=nm[1];    mm[2]=nm[39];   mm[3]=nm[38];   mm[4]=nm[4];
  mm[5]=nm[5];    mm[6]=nm[35];   mm[7]=nm[34];   mm[8]=nm[8];
  mm[9]=nm[53];   mm[10]=nm[19];  mm[11]=nm[18];  mm[12]=nm[56];
  mm[13]=nm[49];  mm[14]=nm[23];  mm[15]=nm[22];  mm[16]=nm[52];
  mm[17]=nm[45];  mm[18]=nm[11];  mm[19]=nm[10];  mm[20]=nm[48];
  mm[21]=nm[41];  mm[22]=nm[15];  mm[23]=nm[14];  mm[24]=nm[44];
}

```

```

mm[25]=nm[25]; mm[26]=nm[63]; mm[27]=nm[62]; mm[28]=nm[28];
mm[29]=nm[29]; mm[30]=nm[59]; mm[31]=nm[58]; mm[32]=nm[32];
mm[33]=nm[33]; mm[34]=nm[7]; mm[35]=nm[6]; mm[36]=nm[36];
mm[37]=nm[37]; mm[38]=nm[3]; mm[39]=nm[2]; mm[40]=nm[40];
mm[41]=nm[21]; mm[42]=nm[51]; mm[43]=nm[50]; mm[44]=nm[24];
mm[45]=nm[17]; mm[46]=nm[55]; mm[47]=nm[54]; mm[48]=nm[20];
mm[49]=nm[13]; mm[50]=nm[43]; mm[51]=nm[42]; mm[52]=nm[16];
mm[53]=nm[9]; mm[54]=nm[47]; mm[55]=nm[46]; mm[56]=nm[12];
mm[57]=nm[57]; mm[58]=nm[31]; mm[59]=nm[30]; mm[60]=nm[60];
mm[61]=nm[61]; mm[62]=nm[27]; mm[63]=nm[26]; mm[64]=nm[64];
}
/**/
/** Transformation with Pan-magic Line Rotations from 1 into 8x8 */
void trnsfpd(short x){
short m,m8,l,l8,n,nn;
for(m=0;m<7;m++){m8=m*8;
for(n=0;n<64;n=n+8){
for(nn=1;nn<8;nn++){trn[x+m8+8][n+nn]=trn[x+m8][n+nn+1];}
trn[x+m8+8][n+8]=trn[x+m8][n+1];
}}
for(l=0;l<8;l++){l8=l*8;
for(m=0;m<7;m++){
for(n=0;n<56;n=n+8){
for(nn=1;nn<9;nn++){trn[x+l8+m+1][n+nn]=trn[x+l8+m][n+nn+8];}}
for(nn=1;nn<9;nn++){trn[x+l8+m+1][56+nn]=trn[x+l8+m][nn];}
}}
}
/**/
/* End_Of_File */

```

Registration procedure of the produced solutions may be rather difficult for you to understand what it should do in my program. We must check and remove any repetition of the same answer and must register only a new answer every time.

It is also the critical problem where in our memory space we have to save them.

We must be smart enough to manage our computer to work well whenever we deal with a lot of data. Sorting the big object data for our smart listing is always a hard job.

## #8. Result Report

You will surely have the same report as follows, if you make the same experiment with the program dictated by me as above.

```

*** Compose 'Multiple' type of 'Composite & Complete' Magic Squares 8x8 ***
** by New Method Using All Possible View-Forms of Developed Extra-Cubic **
** Objects of Order 2^6 and 'Do-it-After-the-Model' Transformation **

** List of Solution Counts according to the Values of N1: **
( 1: 720) ( 2: 720) ( 3: 720) ( 4: 720) ( 5: 720) ( 6: 720) ( 7: 720) ( 8: 720)
( 9: 720) (10: 720) (11: 720) (12: 720) (13: 720) (14: 720) (15: 720) (16: 720)
(17: 720) (18: 720) (19: 720) (20: 720) (21: 720) (22: 720) (23: 720) (24: 720)
(25: 720) (26: 720) (27: 720) (28: 720) (29: 720) (30: 720) (31: 720) (32: 720)
(33: 720) (34: 720) (35: 720) (36: 720) (37: 720) (38: 720) (39: 720) (40: 720)
(41: 720) (42: 720) (43: 720) (44: 720) (45: 720) (46: 720) (47: 720) (48: 720)
(49: 720) (50: 720) (51: 720) (52: 720) (53: 720) (54: 720) (55: 720) (56: 720)
(57: 720) (58: 720) (59: 720) (60: 720) (61: 720) (62: 720) (63: 720) (64: 720)
Sum = 46080

** Transform 'Multiple' type into 'Standard' type of 'C&C' Magic Squares 8x8
by the Pan-magic Line Rotation Plus Transformation System from 1 into 4x64 **
... Monitoring Total Solution Counts in Every Step:
[ 1: 22896] [ 2: 45792] [ 3: 68784] [ 4: 91776]
[ 5: 112560] [ 6: 133344] [ 7: 154224] [ 8: 175104]
[ 9: 191152] [10: 207200] [11: 223312] [12: 239424]
[13: 253264] [14: 267104] [15: 281008] [16: 294912]

```

[17: 304064] [18: 313216] [19: 322400] [20: 331584]  
 [21: 338528] [22: 345472] [23: 352448] [24: 359424]  
 [25: 361728] [26: 364032] [27: 366336] [28: 368640]  
 [29: 368640] [30: 368640] [31: 368640] [32: 368640]

... Sorting the Solution Data for the smart List of Standard Set.  
 You may have to wait for a long time, I am afraid.

**\*\* List of Standard Solutions: Standard Type of 'Composite & Complete' MS88 \*\***

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

29/ 1 63 4 62 8 58 5 59 32 34 29 35 25 39 28 38 9 55 12 54 16 50 13 51 24 42 21 43 17 47 20 46 57 7 60 6 64 2 61 3 40 26 37 27 33 31 36 30 49 15 52 14 56 10 53 11 48 18 45 19 41 23 44 22	30/ 1 63 4 62 8 58 5 59 32 34 29 35 25 39 28 38 49 15 52 14 56 10 53 11 24 42 21 43 17 47 20 46 57 7 60 6 64 2 61 3 40 26 37 27 33 31 36 30 9 55 12 54 16 50 13 51 48 18 45 19 41 23 44 22	31/ 1 63 4 62 8 58 5 59 32 34 29 35 25 39 28 38 17 47 20 46 24 42 21 43 16 50 13 51 9 55 12 54 57 7 60 6 64 2 61 3 40 26 37 27 33 31 36 30 41 23 44 22 48 18 45 19 56 10 53 11 49 15 52 14	32/ 1 63 4 62 8 58 5 59 32 34 29 35 25 39 28 38 41 23 44 22 48 18 45 19 16 50 13 51 9 55 12 54 57 7 60 6 64 2 61 3 40 26 37 27 33 31 36 30 17 47 20 46 24 42 21 43 56 10 53 11 49 15 52 14
33/ 1 63 4 62 8 58 5 59 24 42 21 43 17 47 20 46 25 39 28 38 32 34 29 35 56 10 53 11 49 15 52 14 57 7 60 6 64 2 61 3 48 18 45 19 41 23 44 22 33 31 36 30 40 26 37 27 16 50 13 51 9 55 12 54	34/ 1 63 4 62 8 58 5 59 24 42 21 43 17 47 20 46 33 31 36 30 40 26 37 27 56 10 53 11 49 15 52 14 57 7 60 6 64 2 61 3 48 18 45 19 41 23 44 22 25 39 28 38 32 34 29 35 16 50 13 51 9 55 12 54	35/ 1 63 4 62 8 58 5 59 24 42 21 43 17 47 20 46 9 55 12 54 16 50 13 51 40 26 37 27 33 31 36 30 57 7 60 6 64 2 61 3 48 18 45 19 41 23 44 22 49 15 52 14 56 10 53 11 32 34 29 35 25 39 28 38	36/ 1 63 4 62 8 58 5 59 24 42 21 43 17 47 20 46 49 15 52 14 56 10 53 11 40 26 37 27 33 31 36 30 57 7 60 6 64 2 61 3 48 18 45 19 41 23 44 22 9 55 12 54 16 50 13 51 32 34 29 35 25 39 28 38
5757/ 1 62 9 54 15 52 7 60 48 19 40 27 34 29 42 21 2 61 10 53 16 51 8 59 32 35 24 43 18 45 26 37 50 13 58 5 64 3 56 11 31 36 23 44 17 46 25 38 49 14 57 6 63 4 55 12 47 20 39 28 33 30 41 22	5758/ 1 62 9 54 15 52 7 60 48 19 40 27 34 29 42 21 49 14 57 6 63 4 55 12 32 35 24 43 18 45 26 37 50 13 58 5 64 3 56 11 31 36 23 44 17 46 25 38 2 61 10 53 16 51 8 59 47 20 39 28 33 30 41 22	5759/ 1 62 9 54 15 52 7 60 48 19 40 27 34 29 42 21 18 45 26 37 32 35 24 43 16 51 8 59 2 61 10 53 50 13 58 5 64 3 56 11 31 36 23 44 17 46 25 38 33 30 41 22 47 20 39 28 63 4 55 12 49 14 57 6	5760/ 1 62 9 54 15 52 7 60 48 19 40 27 34 29 42 21 33 30 41 22 47 20 39 28 16 51 8 59 2 61 10 53 50 13 58 5 64 3 56 11 31 36 23 44 17 46 25 38 18 45 26 37 32 35 24 43 63 4 55 12 49 14 57 6
5761/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 17 46 25 38 31 36 23 44 63 4 55 12 49 14 57 6 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 34 29 42 21 48 19 40 27 16 51 8 59 2 61 10 53	5762/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 34 29 42 21 48 19 40 27 63 4 55 12 49 14 57 6 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 17 46 25 38 31 36 23 44 16 51 8 59 2 61 10 53	5763/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 2 61 10 53 16 51 8 59 48 19 40 27 34 29 42 21 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 49 14 57 6 63 4 55 12 31 36 23 44 17 46 25 38	5764/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 49 14 57 6 63 4 55 12 48 19 40 27 34 29 42 21 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 2 61 10 53 16 51 8 59 31 36 23 44 17 46 25 38
5765/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 2 61 10 53 16 51 8 59 31 36 23 44 17 46 25 38 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 49 14 57 6 63 4 55 12 48 19 40 27 34 29 42 21	5766/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 49 14 57 6 63 4 55 12 31 36 23 44 17 46 25 38 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 2 61 10 53 16 51 8 59 48 19 40 27 34 29 42 21	5767/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 17 46 25 38 31 36 23 44 16 51 8 59 2 61 10 53 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 34 29 42 21 48 19 40 27 63 4 55 12 49 14 57 6	5768/ 1 62 9 54 15 52 7 60 47 20 39 28 33 30 41 22 34 29 42 21 48 19 40 27 16 51 8 59 2 61 10 53 50 13 58 5 64 3 56 11 32 35 24 43 18 45 26 37 17 46 25 38 31 36 23 44 63 4 55 12 49 14 57 6
23037/ 1 40 17 44 29 60 13 56 30 59 14 55 2 39 18 43 3 38 19 42 31 58 15 54 32 57 16 53 4 37 20 41 36 5 52 9 64 25 48 21 63 26 47 22 35 6 51 10 34 7 50 11 62 27 46 23 61 28 45 24 33 8 49 12	23038/ 1 40 17 44 29 60 13 56 30 59 14 55 2 39 18 43 34 7 50 11 62 27 46 23 32 57 16 53 4 37 20 41 36 5 52 9 64 25 48 21 63 26 47 22 35 6 51 10 3 38 19 42 31 58 15 54 61 28 45 24 33 8 49 12	23039/ 1 40 17 44 29 60 13 56 30 59 14 55 2 39 18 43 4 37 20 41 32 57 16 53 31 58 15 54 3 38 19 42 36 5 52 9 64 25 48 21 63 26 47 22 35 6 51 10 33 8 49 12 61 28 45 24 62 27 46 23 34 7 50 11	23040/ 1 40 17 44 29 60 13 56 30 59 14 55 2 39 18 43 33 8 49 12 61 28 45 24 31 58 15 54 3 38 19 42 36 5 52 9 64 25 48 21 63 26 47 22 35 6 51 10 4 37 20 41 32 57 16 53 62 27 46 23 34 7 50 11
23041/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 26 40 28 38 31 33 29 35 47 17 45 19 42 24 44 22 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 34 32 36 30 39 25 37 27 23 41 21 43 18 48 20 46	23042/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 34 32 36 30 39 25 37 27 47 17 45 19 42 24 44 22 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 26 40 28 38 31 33 29 35 23 41 21 43 18 48 20 46	23043/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 18 48 20 46 23 41 21 43 39 25 37 27 34 32 36 30 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 42 24 44 22 47 17 45 19 31 33 29 35 26 40 28 38	23044/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 42 24 44 22 47 17 45 19 39 25 37 27 34 32 36 30 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 18 48 20 46 23 41 21 43 31 33 29 35 26 40 28 38
23045/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 18 48 20 46 23 41 21 43 31 33 29 35 26 40 28 38 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 42 24 44 22 47 17 45 19 39 25 37 27 34 32 36 30	23046/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 42 24 44 22 47 17 45 19 31 33 29 35 26 40 28 38 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 18 48 20 46 23 41 21 43 39 25 37 27 34 32 36 30	23047/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 26 40 28 38 31 33 29 35 23 41 21 43 18 48 20 46 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 34 32 36 30 39 25 37 27 47 17 45 19 42 24 44 22	23048/ 2 64 4 62 7 57 5 59 55 9 53 11 50 16 52 14 34 32 36 30 39 25 37 27 23 41 21 43 18 48 20 46 58 8 60 6 63 1 61 3 15 49 13 51 10 56 12 54 26 40 28 38 31 33 29 35 47 17 45 19 42 24 44 22

```

. . . . . 46077/ 46078/ 46079/ 46080/
2 39 18 43 30 59 14 55 2 39 18 43 30 59 14 55 2 39 18 43 30 59 14 55 2 39 18 43 30 59 14 55
29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44
3 38 19 42 31 58 15 54 34 7 50 11 62 27 46 23 4 37 20 41 32 57 16 53 33 8 49 12 61 28 45 24
32 57 16 53 4 37 20 41 32 57 16 53 4 37 20 41 31 58 15 54 3 38 19 42 31 58 15 54 3 38 19 42
35 6 51 10 63 26 47 22 35 6 51 10 63 26 47 22 35 6 51 10 63 26 47 22 35 6 51 10 63 26 47 22
64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9
34 7 50 11 62 27 46 23 3 38 19 42 31 58 15 54 33 8 49 12 61 28 45 24 4 37 20 41 32 57 16 53
61 28 45 24 33 8 49 12 61 28 45 24 33 8 49 12 62 27 46 23 34 7 50 11 62 27 46 23 34 7 50 11

. . . . . 46081/ 46082/ 46083/ 46084/
3 64 4 63 6 57 5 58 3 64 4 63 6 57 5 58 3 64 4 63 6 57 5 58 3 64 4 63 6 57 5 58
54 9 53 10 51 16 52 15 54 9 53 10 51 16 52 15 54 9 53 10 51 16 52 15 54 9 53 10 51 16 52 15
27 40 28 39 30 33 29 34 35 32 36 31 38 25 37 26 19 48 20 47 22 41 21 42 43 24 44 23 46 17 45 18
46 17 45 18 43 24 44 23 46 17 45 18 43 24 44 23 38 25 37 26 35 32 36 31 38 25 37 26 35 32 36 31
59 8 60 7 62 1 61 2 59 8 60 7 62 1 61 2 59 8 60 7 62 1 61 2 59 8 60 7 62 1 61 2
14 49 13 50 11 56 12 55 14 49 13 50 11 56 12 55 14 49 13 50 11 56 12 55 14 49 13 50 11 56 12 55
35 32 36 31 38 25 37 26 27 40 28 39 30 33 29 34 43 24 44 23 46 17 45 18 19 48 20 47 22 41 21 42
22 41 21 42 19 48 20 47 22 41 21 42 19 48 20 47 30 33 29 34 27 40 28 39 30 33 29 34 27 40 28 39

. . . . . 68733/ 68734/ 68735/ 68736/
3 38 19 42 31 58 15 54 3 38 19 42 31 58 15 54 3 38 19 42 31 58 15 54 3 38 19 42 31 58 15 54
29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44 29 60 13 56 1 40 17 44
2 39 18 43 30 59 14 55 35 6 51 10 63 26 47 22 4 37 20 41 32 57 16 53 33 8 49 12 61 28 45 24
32 57 16 53 4 37 20 41 32 57 16 53 4 37 20 41 30 59 14 55 2 39 18 43 30 59 14 55 2 39 18 43
34 7 50 11 62 27 46 23 34 7 50 11 62 27 46 23 34 7 50 11 62 27 46 23 34 7 50 11 62 27 46 23
64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9 64 25 48 21 36 5 52 9
35 6 51 10 63 26 47 22 2 39 18 43 30 59 14 55 33 8 49 12 61 28 45 24 4 37 20 41 32 57 16 53
61 28 45 24 33 8 49 12 61 28 45 24 33 8 49 12 63 26 47 22 35 6 51 10 63 26 47 22 35 6 51 10

. . . . . 68737/ 68738/ 68739/ 68740/
4 64 3 63 5 57 6 58 4 64 3 63 5 57 6 58 4 64 3 63 5 57 6 58 4 64 3 63 5 57 6 58
53 9 54 10 52 16 51 15 53 9 54 10 52 16 51 15 53 9 54 10 52 16 51 15 53 9 54 10 52 16 51 15
28 40 27 39 29 33 30 34 36 32 35 31 37 25 38 26 20 48 19 47 21 41 22 42 44 24 43 23 45 17 46 18
45 17 46 18 44 24 43 23 45 17 46 18 44 24 43 23 37 25 38 26 36 32 35 31 37 25 38 26 36 32 35 31
60 8 59 7 61 1 62 2 60 8 59 7 61 1 62 2 60 8 59 7 61 1 62 2 60 8 59 7 61 1 62 2
13 49 14 50 12 56 11 55 13 49 14 50 12 56 11 55 13 49 14 50 12 56 11 55 13 49 14 50 12 56 11 55
36 32 35 31 37 25 38 26 28 40 27 39 29 33 30 34 44 24 43 23 45 17 46 18 20 48 19 47 21 41 22 42
21 41 22 42 20 48 19 47 21 41 22 42 20 48 19 47 29 33 30 34 28 40 27 39 29 33 30 34 28 40 27 39

. . . . . 368629/ 368630/ 368631/ 368632/
30 37 26 55 12 51 16 33 30 37 26 55 12 51 16 33 30 37 26 55 12 51 16 33 30 37 26 55 12 51 16 33
4 59 8 41 22 45 18 63 4 59 8 41 22 45 18 63 3 60 7 42 21 46 17 64 3 60 7 42 21 46 17 64
21 46 17 64 3 60 7 42 62 5 58 23 44 19 48 1 22 45 18 63 4 59 8 41 61 6 57 24 43 20 47 2
11 52 15 34 29 38 25 56 11 52 15 34 29 38 25 56 11 52 15 34 29 38 25 56 11 52 15 34 29 38 25 56
53 14 49 32 35 28 39 10 53 14 49 32 35 28 39 10 53 14 49 32 35 28 39 10 53 14 49 32 35 28 39 10
43 20 47 2 61 6 57 24 43 20 47 2 61 6 57 24 44 19 48 1 62 5 58 23 44 19 48 1 62 5 58 23
62 5 58 23 44 19 48 1 21 46 17 64 3 60 7 42 61 6 57 24 43 20 47 2 22 45 18 63 4 59 8 41
36 27 40 9 54 13 50 31 36 27 40 9 54 13 50 31 36 27 40 9 54 13 50 31 36 27 40 9 54 13 50 31

. . . . . 368633/ 368634/ 368635/ 368636/
30 37 24 47 20 43 26 33 30 37 24 47 20 43 26 33 30 37 24 47 20 43 26 33 30 37 24 47 20 43 26 33
4 59 10 49 14 53 8 63 4 59 10 49 14 53 8 63 3 60 9 50 13 54 7 64 3 60 9 50 13 54 7 64
13 54 7 64 3 60 9 50 62 5 56 15 52 11 58 1 14 53 8 63 4 59 10 49 61 6 55 16 51 12 57 2
19 44 25 34 29 38 23 48 19 44 25 34 29 38 23 48 19 44 25 34 29 38 23 48 19 44 25 34 29 38 23 48
45 22 39 32 35 28 41 18 45 22 39 32 35 28 41 18 45 22 39 32 35 28 41 18 45 22 39 32 35 28 41 18
51 12 57 2 61 6 55 16 51 12 57 2 61 6 55 16 52 11 58 1 62 5 56 15 52 11 58 1 62 5 56 15
62 5 56 15 52 11 58 1 13 54 7 64 3 60 9 50 61 6 55 16 51 12 57 2 14 53 8 63 4 59 10 49
36 27 42 17 46 21 40 31 36 27 42 17 46 21 40 31 36 27 42 17 46 21 40 31 36 27 42 17 46 21 40 31

. . . . . 368637/ 368638/ 368639/ 368640/
30 37 26 47 20 43 24 33 30 37 26 47 20 43 24 33 30 37 26 47 20 43 24 33 30 37 26 47 20 43 24 33
4 59 8 49 14 53 10 63 4 59 8 49 14 53 10 63 3 60 7 50 13 54 9 64 3 60 7 50 13 54 9 64
13 54 9 64 3 60 7 50 62 5 58 15 52 11 56 1 14 53 10 63 4 59 8 49 61 6 57 16 51 12 55 2
19 44 23 34 29 38 25 48 19 44 23 34 29 38 25 48 19 44 23 34 29 38 25 48 19 44 23 34 29 38 25 48
45 22 41 32 35 28 39 18 45 22 41 32 35 28 39 18 45 22 41 32 35 28 39 18 45 22 41 32 35 28 39 18
51 12 55 2 61 6 57 16 51 12 55 2 61 6 57 16 52 11 56 1 62 5 58 15 52 11 56 1 62 5 58 15
62 5 58 15 52 11 56 1 13 54 9 64 3 60 7 50 61 6 57 16 51 12 55 2 14 53 10 63 4 59 8 49
36 27 40 17 46 21 42 31 36 27 40 17 46 21 42 31 36 27 40 17 46 21 42 31 36 27 40 17 46 21 42 31

```

[Count = 368640]

\*\* List of Solution Counts according to the Values of N1: \*\*

```

( 1:23040) ( 2:23040) ( 3:22656) ( 4:22656) ( 5:21696) ( 6:21696) ( 7:21312) ( 8:21312)
( 9:18240) (10:18240) (11:17856) (12:17856) (13:16896) (14:16896) (15:16512) (16:16512)
(17: 6528) (18: 6528) (19: 6144) (20: 6144) (21: 5184) (22: 5184) (23: 4800) (24: 4800)
(25: 1728) (26: 1728) (27: 1344) (28: 1344) (29: 384) (30: 384) (31: 0) (32: 0)

```



