

## 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 1. About Some Fundamentals of High-Dimensional Extra-Cubic Objects and their Developed Forms

##### #1. Introduction

I have found something new recently. I suppose it might be surely interesting to you, Magic Square builders and researchers. Now let me introduce you my newest method of composing high-dimensional magic objects and their developed forms in the following several sections.

The idea came up to me when I found how many 'view forms' we must draw for a developed Extra-Cubic object of high dimensions and knew how we could draw them.

As you know we, human being, can draw no single picture of high dimensional Extra-Cubic Object. We cannot catch the whole thing directly at a sight, either. We can only imagine it in our brain by any intellectual method.

But without using any figures it is too difficult for us to think of and build them, so I always tried to prepare different kinds of several 'view-forms' developed into lower dimensions, as you see in the previous chapters.

But I did not yet try to draw many different figures of the same kind.

It is just a conventional habit that we have usually prepared a single Basic Diagram for any object, because it is actually enough for 2- or 3-dimensions.

Then I noticed we must draw many different 'view-forms' of the same kind for our developed objects, as many as we could. Otherwise we could not even define the essential simultaneous equations for our ECO all right at the first stage.

How many pictures can we draw for those, then? How many diagrams do we have to prepare? And how can we draw them? By what program can we make our computer design, draw and print them out all right?

Let's try to solve these rather puzzling problems clearly, shall we?

But I feel I must prepare a few educational lessons for you to understand it well.

I must talk about computer programming only a little.

##### #2. Some Lessons of Computer Programming

I feel sorry to write about computer programming, but we cannot pass without explaining it. Excuse me, please. I will try to explain everything as simply as possible.

I just want you to understand how important the multiple-loops of `for( ... ){ ... }` sentences are, and know the relationship to the Positional Number System.

The following two programs have only single loop of `for( ... ){ ... }` sentence.

```
/**/  
/** Program #1 **/  
void pr41(){  
    short n;  
    printf("\nStudy #41: One Dimensional Serial Numbers\n");  
    for(n=1; n<17; n++){  
        printf("%3d", n);  
    }  
    printf("\n");  
}
```

```

/**/
/* Program #2 */
void pr42(){
short n;
printf("\nStudy #42: Two Dimensional Placement of Numbers:\n");
printf(" /8 Pairs of Self-Complementary Numbers\n");
printf(" ");
for(n=1;n<9;n++){printf("%3d",n);}
printf("\n +");
for(n=1;n<9;n++){printf("%3d",17-n);}
printf("\n ----- \n ");
for(n=1;n<9;n++){printf("%3d",17);}
printf("\n");
}
/**/

```

The outputs of these programs look as follows:

```

/* Result */
Study #41: One Dimensional Serial Numbers
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

Study #42: Two Dimensional Placement of Numbers:
/8 Pairs of Self-Complementary Numbers
  1 2 3 4 5 6 7 8
+) 16 15 14 13 12 11 10 9
-----
 17 17 17 17 17 17 17 17

```

The next program has double loops of `for( ... ){ ... }` sentences for the first time. It acts like this:

```

/**/
/* Program #3 */
void pr43(){
short m, n;
printf("\nStudy #43: Two Dimensional Placement of Numbers:\n");
printf(" /Two Dimensional Square of Order 4\n");
for(m=0;m<4;m++){
printf(" ");
for(n=0;n<4;n++){
printf("%3d",m*4+n+1);
}
printf("\n");
}
}
/**/
/* Result */
Study #43: Two Dimensional Placement of Numbers:
/Two Dimensional Square of Order 4
 1 2 3 4
 5 6 7 8
 9 10 11 12
13 14 15 16

```

Though it only writes the serial numbers from 1 to 16 just 4 by 4 in all lines, the result looks like a 'Basic Diagram' for any type of Magic Square of order 4.

This picture can be used for a 'Position Names' diagram or a 'Prototype' square. Each natural number is made up by calculating  $(m*4+n+1)$  in this program.

Yes. Double loops make every number by the Positional Writing System of Numbers of the Base 4.

In fact it is the generating engine of 'N4i' (though every number is written as (N4i+1) here in our classical notation).

The next program will print out the 'Basic Diagrams' for MS44 most likely.

```

/* Program #4 */
void pr44(){
short d0, d1, d;
printf("\nStudy #44: Two Dimensional Square of Order 4\n");
printf("    Classic/    Positional Parameters/\n");
for(d0=0; d0<4; d0++){
printf(" ");
for(d1=0; d1<4; d1++){printf("%3d", d0*4+d1+1); } //Change
printf(" ");
/**/
for(d1=0; d1<4; d1++){printf(" (%d, %d)", d0, d1); } //Change
printf("\n");
}
/**/
printf(" Mathematical /           N4i /           H/D4i /L\n");
for(d0=0; d0<4; d0++){
printf(" ");
for(d1=0; d1<4; d1++){printf("%3d", d0*4+d1); } //Change
printf(" ");
/**/
for(d1=0; d1<4; d1++){
printf(" %d%d", d0, d1); } //Change
printf(" ");
/**/
printf("%2d%2d%2d%2d ", d0, d0, d0, d0); //Change
for(d1=0; d1<4; d1++){
printf("%2d", d1); } //Change
printf("\n");
}
}
/**/

```

Study #44: Two Dimensional Square of Order 4

	Classic/	Positional	Coordinates/
1 2 3 4	(0, 0)	(0, 1)	(0, 2) (0, 3)
5 6 7 8	(1, 0)	(1, 1)	(1, 2) (1, 3)
9 10 11 12	(2, 0)	(2, 1)	(2, 2) (2, 3)
13 14 15 16	(3, 0)	(3, 1)	(3, 2) (3, 3)
<b>Mathematical /</b>	<b>N4i /</b>	<b>H/D4i /L</b>	
0 1 2 3	00 01 02 03	0 0 0 0	0 1 2 3
4 5 6 7	10 11 12 13	1 1 1 1	0 1 2 3
8 9 10 11	20 21 22 23	2 2 2 2	0 1 2 3
12 13 14 15	30 31 32 33	3 3 3 3	0 1 2 3
	(Notation)	(Decomposition)	

You could surely know what I mean about 'N4i generator' of double loops. It really makes every number by (d0\*4+d1). It calculates no division here at all.

Step forward to the next lesson I invented for you to understand the effect of a little 'change' or 'modification' of the former program.

```

/* Program #5 */
void pr45(){

```

```

short d0, d1, d;
printf("\nStudy #45: Another View of 2 Dimensional Square of Order 4\n");
printf("    Classic/    Positional Parameters/\n");
for(d0=0; d0<4; d0++){
    printf("    ");
    for(d1=0; d1<4; d1++){printf("%3d", d1*4+d0+1); } //Change
    printf("    ");
    /**/
    for(d1=0; d1<4; d1++){printf(" (%d, %d)", d1, d0); } //Change
    printf("\n");
}
/**/
printf(" Mathematical /           N4i /           H/D4i /L\n");
for(d0=0; d0<4; d0++){
    printf("    ");
    for(d1=0; d1<4; d1++){printf("%3d", d1*4+d0); } //Change
    printf("    ");
    /**/
    for(d1=0; d1<4; d1++){
        printf(" %d%d", d1, d0); } //Change
    printf("    ");
    /**/ //Change All
    for(d1=0; d1<4; d1++){
        printf("%2d", d1); }
    printf(" %2d%2d%2d%2d", d0, d0, d0, d0);
    printf("\n");
}
}
/**/

```

Study #45: Another View of 2 Dimensional Square of Order 4

	Classic/	Positional	Coordinates/
1	5 9 13	(0, 0)	(1, 0) (2, 0) (3, 0)
2	6 10 14	(0, 1)	(1, 1) (2, 1) (3, 1)
3	7 11 15	(0, 2)	(1, 2) (2, 2) (3, 2)
4	8 12 16	(0, 3)	(1, 3) (2, 3) (3, 3)

	Mathematical /	N4i /	H/D4i /L
0	4 8 12	00 10 20 30	0 1 2 3 0 0 0 0
1	5 9 13	01 11 21 31	0 1 2 3 1 1 1 1
2	6 10 14	02 12 22 32	0 1 2 3 2 2 2 2
3	7 11 15	03 13 23 33	0 1 2 3 3 3 3 3

You see 'symmetrical exchange' occurs with respect to one of the primary diagonals, or 'mirror reflection' is made by these modifications. This result is also one of the 'Basic Diagrams' and indicates any way of 'Basic Transformation'.

Isn't it interesting?

### #3. The Second Lessons for 4-Dimensional ECO of Order 2

Let's go forward to the next step where we can deal with the 4-dimensional Extra-Cubic Object of order 2 by multiple loops of `for( ... ){ ... }`.

Won't you read the following simple program of mine?

```

/* Program #6 */
void pr46(){
short d0, d1, d2, d3;
printf("\nStudy #46: Four Dimensional Extra-Cubic Object 2\n");
printf(" /Simple View Diagram of 2x2^3 Extra-Cubic Object\n");
for(d0=0; d0<2; d0++){

```

```

for(d1=0; d1<2; d1++){
  for(d2=0; d2<2; d2++){
    if(d2>0){printf("  ");}
    for(d3=0; d3<2; d3++){printf("%6d", ((d0*2+d1)*2+d2)*2+d3+1);}
    printf("\n");
  }
}
printf("\n");
}
}
/* result */

```

#### Study #46: Four Dimensional Extra-Cubic Object 2

/Simple View Diagram of  $2 \times 2^3$  Extra-Cubic Object

```

1-----2
| 3---+---4
5--|---6 |
  7-----8

9-----10
| 11---+---12
13--|---14 |
  15-----16

```

It uses four-time loops of `for(...){...}` sentences and generates binary numbers.

$((d0*2+d1)*2+d2)*2+d3+1$  shows a typical calculation of binary numbers.

Though it only writes the serial numbers from 1 to 16 just 2 by 2 in every line, it looks like a 'Basic Diagram' for our developed  $ECO2^4$  while it is well designed with some appropriate spaces and carriage returns.

The next program will print out the Basic Diagrams most likely.

```

/* Program #7 */
void pr47(){
  short d0, d1, d2, d3;
  printf("Study #47: Four Dimensional Extra-Cubic Object of Order 2\n");
  for(d0=0; d0<2; d0++){
    printf(" D%d/\n", d0);
    for(d1=0; d1<2; d1++){
      for(d2=0; d2<2; d2++){
        if(d2>0){printf("  ");}
        for(d3=0; d3<2; d3++){printf("%4d ", ((d0*2+d1)*2+d2)*2+d3+1);}
        printf(" ");
        /**/
        for(d3=0; d3<2; d3++){printf("%4d ", ((d0*2+d1)*2+d2)*2+d3);}
        printf(" ");
        /**/
        for(d3=0; d3<2; d3++){printf(" %d%d%d%d", d0, d1, d2, d3);}
        printf(" ");
        /**/
        printf("%3d%3d %3d%3d %3d%3d ", d0, d0, d1, d1, d2, d2);
        for(d3=0; d3<2; d3++){
          printf("%3d", d3); }
        printf("\n");
      }
    }
  }
}
printf(" /Classical /Mathematical /N2i /D2i \n");
}

```

```

/**/
/* Result */
Study #47: Four Dimensional Extra-Cubic Object of Order 2
D0/
1----2      0   1      0000 0001      0 0      0 0      0 0      0 1
| 3---+4      2   3      0010 0011      0 0      0 0      1 1      0 1
5-|-6 |      4   5      0100 0101      0 0      1 1      0 0      0 1
  7----8      6   7      0110 0111      0 0      1 1      1 1      0 1
D1/
9---10      8   9      1000 1001      1 1      0 0      0 0      0 1
|11---+12     10  11      1010 1011      1 1      0 0      1 1      0 1
13-|-14 |     12  13      1100 1101      1 1      1 1      0 0      0 1
  15---16     14  15      1110 1111      1 1      1 1      1 1      0 1
/Classi c   /Mathemati cal   /N2i           /D2i

```

Let's make some experiments by changing the former program just a little.

```

/* Program #8 */
void pr48(){
short d0, d1, d2, d3;
printf("\nStudy #48: Another Vi ew of 4 Di mensi onal Extra-Cubi c Object 2^4\n");
for(d0=0; d0<2; d0++){
printf(" D%d           /N2i           /D2i /\n", d0);
for(d1=0; d1<2; d1++){
for(d2=0; d2<2; d2++){
if(d2>0){printf(" ");}
for(d3=0; d3<2; d3++){printf("%4d ", ((d3*2+d2)*2+d1)*2+d0+1);}
printf(" ");
/**/
for(d3=0; d3<2; d3++){printf(" %d%d%d%d", d3, d2, d1, d0);}
printf(" ");
/**/
for(d3=0; d3<2; d3++){
printf("%3d", d3);}
printf(" %3d%3d %3d%3d %3d%3d", d2, d2, d1, d1, d0, d0);
printf("\n");
}
}
}
}
/**/
/* Result */

```

```

Study #48: Another Vi ew of 4 Di mensi onal Extra-Cubi c Object 2^4
D0           /N2i           /D2i /
1----9      0000 1000      0 1      0 0      0 0      0 0
| 5---+13     0100 1100      0 1      1 1      0 0      0 0
3-|-11 |     0010 1010      0 1      0 0      1 1      0 0
  7---15     0110 1110      0 1      1 1      1 1      0 0
D1           /N2i           /D2i /
2---10      0001 1001      0 1      0 0      0 0      1 1
| 6---+14     0101 1101      0 1      1 1      0 0      1 1
4-|-12 |     0011 1011      0 1      0 0      1 1      1 1
  8---16     0111 1111      0 1      1 1      1 1      1 1

```

Could you find what parts of the program I changed? Right. I changed some orders of variables d0, d1, d2 and d3 in a few sentences at the same time.

The result is quite interesting. Another 'view-diagram' comes out for the same

ECO2<sup>4</sup>, and nothing else could come at all. Why!

```

/* Program #9 */
void pr49(){
short d0, d1, d2, d3;
printf("\nStudy #49: Another View of 4 Dimensional Extra-Cubic Object 2x2^3\n");
for(d0=0; d0<2; d0++){
printf(" D%d          /N2i          /D2i /\n", d0);
for(d1=0; d1<2; d1++){
for(d2=0; d2<2; d2++){
if(d2>0){printf(" ");}
for(d3=1; d3>=0; d3--){printf("%5d", ((d0*2+d1)*2+d2)*2+d3+1);}
printf(" ");
/**/
for(d3=1; d3>=0; d3--){printf(" %d%d%d%d", d0, d1, d2, d3);}
printf(" ");
/**/
printf("%3d%3d %3d%3d %3d%3d ", d0, d0, d1, d1, d2, d2);
for(d3=1; d3>=0; d3--){
printf("%3d", d3);}
printf("\n");
}
}
}
}
}
/**/
/* result */

```

```

Study #49: Another View of 4 Dimensional Extra-Cubic Object 2x23
D0          /N2i          /D2i /
 2   1   0001  0000   0 0   0 0   0 0   1 0
 4   3   0011  0010   0 0   0 0   1 1   1 0
 6   5   0101  0100   0 0   1 1   0 0   1 0
 8   7   0111  0110   0 0   1 1   1 1   1 0
D1          /N2i          /D2i /
10  9   1001  1000   1 1   0 0   0 0   1 0
12 11   1011  1010   1 1   0 0   1 1   1 0
14 13   1101  1100   1 1   1 1   0 0   1 0
16 15   1111  1110   1 1   1 1   1 1   1 0

```

Do you understand the meaning of another experiment changing the original program a little? I used the count-down type of conditions in some `for(---){ ... }` sentences, such as `for(d3=1; d3>=0; d3--){ ... }`.

Another 'view-diagram' comes out, and nothing else could come at all.

These new comers look quite different from each other, but they are only different view-forms of the same ECO2<sup>4</sup>, as different 'faces' of the same 'body'.

They don't mean any different bodies or objects at all.

Why don't you examine what 4 numbers come and sit next to the origin where the 4 axes meet on? You can surely know every number will come all right.

#### #4. New lessons for Higher Dimensional Extra-Cubic Object

Let's go forward to the next experiment and see what makes by the six-time loops of `for( ... ){ ... }` sentences.

Why don't you try to make any 'view forms' of ECO2<sup>6</sup> by those loops?

```

/**/
/* Program #10 */

```

```

void pr66(){
short d0, d1, d2, d3, d4, d5, m;
printf("\nP66: Six Dimensional Extra-Cubic Object 2^6=8x2^3\n");
printf("Classical /N2i /D2i\n");
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++){
if(d4==1){printf(" ");}
for(d5=0; d5<2; d5++){
m((((d0*2+d1)*2+d2)*2+d3)*2+d4)*2+d5; printf("%3d ", m+1);}
for(d5=0; d5<2; d5++){printf(" %d%d%d%d%d", d0, d1, d2, d3, d4, d5);}
printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
d0, d0, d1, d1, d2, d2, d3, d3, d4, d4, 0, 1);
printf("\n");
}}
printf("\n");
}}
}
/**/
printf("Another View:\n");
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++){
if(d4==1){printf(" ");}
for(d5=0; d5<2; d5++){
m((((d0*2+d5)*2+d3)*2+d4)*2+d1)*2+d2; //Changed
printf("%3d ", m+1);}
for(d5=0; d5<2; d5++){printf(" %d%d%d%d%d", d0, d5, d3, d4, d1, d2);}
printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
d0, d0, 0, 1, d3, d3, d4, d4, d1, d1, d2, d2); //Changed
printf("\n");
}}
printf("\n");
}}
}
/**/
printf("Another View:\n");
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++){
if(d4==1){printf(" ");}
for(d5=1; d5>=0; d5--){
m((((d0*2+d1)*2+d2)*2+d3)*2+d4)*2+d5; //Changed
printf("%3d ", m+1);}
for(d5=1; d5>=0; d5--){printf(" %d%d%d%d%d", d0, d1, d2, d3, d4, d5);}
printf("%5d%3d%5d%3d%5d%3d%5d%3d%5d%3d",
d0, d0, d1, d1, d2, d2, d3, d3, d4, d4, 1, 0); //Changed
printf("\n");
}}
printf("\n");
}}
}
}
/**/

```

/\* Result \*/

Study #66: Six Dimensional Extra-Cubic Object  $2^6=8x2^3$

Classi c/	/N2i	/D2i							
1----2	000000 000001	0--0	0 0	0 0	0 0	0 0	0 0	0 0	0 1
3---+4	000010 000011	0+-0	0 0	0 0	0 0	0 0	1 1	0 1	
5- --6	000100 000101	0- 0	0 0	0 0	1 1	1 1	0 0	0 1	
7-----8	000110 000111	0--0	0 0	0 0	1 1	1 1	1 1	0 1	
9---10	001000 001001	0 0	0 0	1 1	0 0	0 0	0 0	0 1	
11---+12	001010 001011	0 0	0 0	1 1	0 0	1 1	1 1	0 1	
13- -14	001100 001101	0 0	0 0	1 1	1 1	1 1	0 0	0 1	
15---16	001110 001111	0 0	0 0	1 1	1 1	1 1	1 1	0 1	
17 18	010000 010001	0 0	1 1	0 0	0 0	0 0	0 0	0 1	
19 20	010010 010011	0 0	1 1	0 0	0 0	1 1	1 1	0 1	
21 22	010100 010101	0 0	1 1	0 0	1 1	0 0	0 0	0 1	
23 24	010110 010111	0 0	1 1	0 0	1 1	1 1	1 1	0 1	
25 26	011000 011001	0 0	1 1	1 1	0 0	0 0	0 0	0 1	
27 28	011010 011011	0 0	1 1	1 1	0 0	1 1	1 1	0 1	
29 30	011100 011101	0 0	1 1	1 1	1 1	1 1	0 0	0 1	
31 32	011110 011111	0 0	1 1	1 1	1 1	1 1	1 1	0 1	
33 34	100000 100001	1 1	0 0	0 0	0 0	0 0	0 0	0 1	
35 36	100010 100011	1 1	0 0	0 0	0 0	1 1	1 1	0 1	
37 38	100100 100101	1 1	0 0	0 0	1 1	0 0	0 0	0 1	
39 40	100110 100111	1 1	0 0	0 0	1 1	1 1	1 1	0 1	
41 42	101000 101001	1 1	0 0	1 1	0 0	0 0	0 0	0 1	
43 44	101010 101011	1 1	0 0	1 1	0 0	1 1	1 1	0 1	
45 46	101100 101101	1 1	0 0	1 1	1 1	1 1	0 0	0 1	
47 48	101110 101111	1 1	0 0	1 1	1 1	1 1	1 1	0 1	
49 50	110000 110001	1 1	1 1	0 0	0 0	0 0	0 0	0 1	
51 52	110010 110011	1 1	1 1	0 0	0 0	1 1	1 1	0 1	
53 54	110100 110101	1 1	1 1	0 0	1 1	0 0	0 0	0 1	
55 56	110110 110111	1 1	1 1	0 0	1 1	1 1	1 1	0 1	
57 58	111000 111001	1 1	1 1	1 1	0 0	0 0	0 0	0 1	
59 60	111010 111011	1 1	1 1	1 1	1 1	0 0	1 1	0 1	
61 62	111100 111101	1 1	1 1	1 1	1 1	1 1	0 0	0 1	
63 64	111110 111111	1 1	1 1	1 1	1 1	1 1	1 1	0 1	

Another Vi ew:

1---17	000000 010000	0 0	0 1	0 0	0 0	0 0	0 0	0 0	
5---+21	000100 010100	0 0	0 1	0 0	1 1	0 0	0 0	0 0	
9- -25	001000 011000	0 0	0 1	1 1	0 0	0 0	0 0	0 0	
13---29	001100 011100	0 0	0 1	1 1	1 1	0 0	0 0	0 0	
2 18	000001 010001	0 0	0 1	0 0	0 0	0 0	0 0	1 1	
6 22	000101 010101	0 0	0 1	0 0	1 1	0 0	0 0	1 1	
10 26	001001 011001	0 0	0 1	1 1	0 0	0 0	0 0	1 1	
14 30	001101 011101	0 0	0 1	1 1	1 1	0 0	0 0	1 1	
3 19	000010 010010	0 0	0 1	0 0	0 0	1 1	0 0	0 0	
7 23	000110 010110	0 0	0 1	0 0	1 1	1 1	0 0	0 0	
11 27	001010 011010	0 0	0 1	1 1	0 0	1 1	0 0	0 0	
15 31	001110 011110	0 0	0 1	1 1	1 1	1 1	0 0	0 0	
4 20	000011 010011	0 0	0 1	0 0	0 0	1 1	1 1	1 1	
8 24	000111 010111	0 0	0 1	0 0	1 1	1 1	1 1	1 1	
12 28	001011 011011	0 0	0 1	1 1	0 0	1 1	1 1	1 1	
16 32	001111 011111	0 0	0 1	1 1	1 1	1 1	1 1	1 1	

33	49	100000	110000	1	1	0	1	0	0	0	0	0	0	0	0	0
37	53	100100	110100	1	1	0	1	0	0	1	1	0	0	0	0	0
41	57	101000	111000	1	1	0	1	1	1	0	0	0	0	0	0	0
45	61	101100	111100	1	1	0	1	1	1	1	1	0	0	0	0	0
34	50	100001	110001	1	1	0	1	0	0	0	0	0	0	0	1	1
38	54	100101	110101	1	1	0	1	0	0	1	1	0	0	0	1	1
42	58	101001	111001	1	1	0	1	1	1	0	0	0	0	0	1	1
46	62	101101	111101	1	1	0	1	1	1	1	1	0	0	0	1	1
35	51	100010	110010	1	1	0	1	0	0	0	0	1	1	0	0	0
39	55	100110	110110	1	1	0	1	0	0	1	1	1	1	0	0	0
43	59	101010	111010	1	1	0	1	1	1	0	0	1	1	0	0	0
47	63	101110	111110	1	1	0	1	1	1	1	1	1	1	0	0	0
36	52	100011	110011	1	1	0	1	0	0	0	0	1	1	1	1	1
40	56	100111	110111	1	1	0	1	0	0	1	1	1	1	1	1	1
44	60	101011	111011	1	1	0	1	1	1	0	0	1	1	1	1	1
48	64	101111	111111	1	1	0	1	1	1	1	1	1	1	1	1	1

Another View:

2	1	000001	000000	0	0	0	0	0	0	0	0	0	0	0	0	1	0	
4	3	000011	000010	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
6	5	000101	000100	0	0	0	0	0	0	1	1	0	0	0	0	1	0	
8	7	000111	000110	0	0	0	0	0	0	1	1	1	1	1	1	1	0	
10	9	001001	001000	0	0	0	0	1	1	0	0	0	0	0	0	1	0	
12	11	001011	001010	0	0	0	0	1	1	0	0	1	1	1	1	1	0	
14	13	001101	001100	0	0	0	0	1	1	1	1	0	0	0	0	1	0	
16	15	001111	001110	0	0	0	0	1	1	1	1	1	1	1	1	1	0	
18	17	010001	010000	0	0	1	1	0	0	0	0	0	0	0	0	1	0	
20	19	010011	010010	0	0	1	1	0	0	0	0	1	1	1	1	1	0	
22	21	010101	010100	0	0	1	1	0	0	1	1	0	0	0	0	1	0	
24	23	010111	010110	0	0	1	1	0	0	1	1	1	1	1	1	1	0	
26	25	011001	011000	0	0	1	1	1	1	0	0	0	0	0	0	1	0	
28	27	011011	011010	0	0	1	1	1	1	0	0	1	1	1	1	1	0	
30	29	011101	011100	0	0	1	1	1	1	1	1	0	0	0	0	1	0	
32	31	011111	011110	0	0	1	1	1	1	1	1	1	1	1	1	1	0	
34	33	100001	100000	1	1	0	0	0	0	0	0	0	0	0	0	1	0	
36	35	100011	100010	1	1	0	0	0	0	0	0	1	1	1	1	1	0	
38	37	100101	100100	1	1	0	0	0	0	1	1	0	0	0	0	1	0	
40	39	100111	100110	1	1	0	0	0	0	1	1	1	1	1	1	1	0	
42	41	101001	101000	1	1	0	0	1	1	0	0	0	0	0	0	1	0	
44	43	101011	101010	1	1	0	0	1	1	0	0	1	1	1	1	1	0	
46	45	101101	101100	1	1	0	0	1	1	1	1	0	0	0	0	1	0	
48	47	101111	101110	1	1	0	0	1	1	1	1	1	1	1	1	1	0	
50	49	110001	110000	1	1	1	1	0	0	0	0	0	0	0	0	1	0	
52	51	110011	110010	1	1	1	1	0	0	0	0	1	1	1	1	1	0	
54	53	110101	110100	1	1	1	1	0	0	1	1	0	0	0	0	1	0	
56	55	110111	110110	1	1	1	1	0	0	1	1	1	1	1	1	1	0	
58	57	111001	111000	1	1	1	1	1	1	0	0	0	0	0	0	1	0	
60	59	111011	111010	1	1	1	1	1	1	0	0	1	1	1	1	1	0	
62	61	111101	111100	1	1	1	1	1	1	1	1	0	0	0	0	1	0	
64	63	111111	111110	1	1	1	1	1	1	1	1	1	1	1	1	1	0	

The result shows our success in making the 'Basic Diagrams' of ECO2<sup>6</sup> and knowing how effective the multiple loops of for(...){...} sentences are.

We also know how effective the 'little changes' of the original program are. They generate some different view forms of the same object successfully.

But what does this success imply? What can we use this for?

And why could we make various view forms of ECO2°?

I suppose one of the reasons is surely in 'the generator program of binary numbers'.

How about the case of Extra-Cubic Object of order 3<sup>4</sup>? Do the four-time loops of `for(...){...}` sentences make any 'view forms' for the developed object?

```
/** Sample of Basic View-Forms of Developed ECO3^4 */
/**/
printf("#1\n");
for(d0=0; d0<3; d0++){
  for(d1=0; d1<3; d1++){
    for(n=d1; n>=0; n--){printf(" ");}
    for(d2=0; d2<3; d2++){
      for(d3=0; d3<3; d3++){
        printf("%2d ",
          ((d0*3+d1)*3+d2)*3+d3+1); //Original
        printf(" ");}
      printf("\n");}}
/**/
printf("#2\n");
for(d0=0; d0<3; d0++){
  for(d1=0; d1<3; d1++){
    for(n=d1; n>=0; n--){printf(" ");}
    for(d2=0; d2<3; d2++){
      for(d3=0; d3<3; d3++){
        printf("%2d ",
          ((d0*3+d1)*3+d3)*3+d2+1); //Changed
        printf(" ");}
      printf("\n");}}
/**/
printf("#3\n");
for(d0=0; d0<3; d0++){
  for(d1=0; d1<3; d1++){
    for(n=d1; n>=0; n--){printf(" ");}
    for(d2=0; d2<3; d2++){
      for(d3=0; d3<3; d3++){
        printf("%2d ",
          ((d2*3+d0)*3+d1)*3+d3+1); //Changed
        printf(" ");}
      printf("\n");}}
/**/
CC=2;
printf("#4\n");
for(d0=0; d0<3; d0++){e0=CC-d0;
  for(d1=0; d1<3; d1++){e1=CC-d1;
    for(n=d1; n>=0; n--){printf(" ");}
    for(d2=0; d2<3; d2++){e2=CC-d2;
      for(d3=0; d3<3; d3++){e3=CC-d3;
        printf("%2d ",
          ((d0*3+d1)*3+d2)*3+e3+1); //Changed
        printf(" ");}
      printf("\n");}}
/**/
printf("#5\n");
for(d0=0; d0<3; d0++){e0=CC-d0;
  for(d1=0; d1<3; d1++){e1=CC-d1;
    for(n=d1; n>=0; n--){printf(" ");}
    for(d2=0; d2<3; d2++){e2=CC-d2;
      for(d3=0; d3<3; d3++){e3=CC-d3;
        printf("%2d ",
          ((e0*3+e1)*3+e2)*3+e3+1); //Changed
```

```

printf(" ");}
printf("\n");}}
/**/
/* Result */
** Sample of Basic View-Forms of Developed Extra-Cubic Object  $3^4$  **

```

#1

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

#2

1	4	7	2	5	8	3	6	9
10	13	16	11	14	17	12	15	18
19	22	25	20	23	26	21	24	27
28	31	34	29	32	35	30	33	36
37	40	43	38	41	44	39	42	45
46	49	52	47	50	53	48	51	54
55	58	61	56	59	62	57	60	63
64	67	70	65	68	71	66	69	72
73	76	79	74	77	80	75	78	81

#3

1	2	3	28	29	30	55	56	57
4	5	6	31	32	33	58	59	60
7	8	9	34	35	36	61	62	63
10	11	12	37	38	39	64	65	66
13	14	15	40	41	42	67	68	69
16	17	18	43	44	45	70	71	72
19	20	21	46	47	48	73	74	75
22	23	24	49	50	51	76	77	78
25	26	27	52	53	54	79	80	81

#4

3	2	1	6	5	4	9	8	7
12	11	10	15	14	13	18	17	16
21	20	19	24	23	22	27	26	25
30	29	28	33	32	31	36	35	34
39	38	37	42	41	40	45	44	43
48	47	46	51	50	49	54	53	52
57	56	55	60	59	58	63	62	61
66	65	64	69	68	67	72	71	70
75	74	73	78	77	76	81	80	79

#5

81	80	79	78	77	76	75	74	73
72	71	70	69	68	67	66	65	64
63	62	61	60	59	58	57	56	55
54	53	52	51	50	49	48	47	46
45	44	43	42	41	40	39	38	37
36	35	34	33	32	31	30	29	28
27	26	25	24	23	22	21	20	19
18	17	16	15	14	13	12	11	10
9	8	7	6	5	4	3	2	1

No doubt the results show our success. They are really 'different view forms' of the same  $ECO3^4$ . Why!

**#5. How many View Forms do we have to draw for High Dimensional ECO?**

Let's consider a little about the next problems now.

- (1) How many different view forms can we draw for the same ECO?
- (2) How many different view forms do we have to prepare for the same ECO at the

definition stage?

Let me take the case of ECO2<sup>4</sup> for instance at first.

1/	/N2i	2/	/N2i	3/	/N2i	4/	/N2i
1	5 0000 0100	1	9 0000 1000	1	9 0000 1000	1	9 0000 1000
2	6 0001 0101	2	10 0001 1001	2	10 0001 1001	3	11 0010 1010
3	7 0010 0110	3	11 0010 1010	5	13 0100 1100	5	13 0100 1100
4	8 0011 0111	4	12 0011 1011	6	14 0101 1101	7	15 0110 1110
9	13 1000 1100	5	13 0100 1100	3	11 0010 1010	2	10 0001 1001
10	14 1001 1101	6	14 0101 1101	4	12 0011 1011	4	12 0011 1011
11	15 1010 1110	7	15 0110 1110	7	15 0110 1110	6	14 0101 1101
12	16 1011 1111	8	16 0111 1111	8	16 0111 1111	8	16 0111 1111
.....							

How many different Basic Diagrams can we draw for the same ECO2<sup>4</sup>?

Watch the next numbers adjacent to the origin n1. They must be n2, n3, n5 or n9. These 4 numbers will appear in different 24 ways, because  $4P_4=4 \times 3 \times 2 \times 1=24$ . Therefore we can draw 24 different Basic Diagrams in all for the same ECO. They always have n1=1 on the left top of the first little cube of order 2.

How can we draw so many Basic Diagrams as 24 all right?

How about adopting our new technique using 4-time loops of for(...){...} with many modifications? Permutation of {d0, d1, d2, d3} is equal to  $24=4 \times 3 \times 2 \times 1$ .

But, do we really have to prepare so many diagrams for the definition stage? We have to do so only to get the correct set of 'simultaneous equations' as follows.

1/d0	/d1	n1+n2+n3+n4=C	n9+n10+n11+n12=C
1----- 2	9-----10	n1+n2+n5+n6=C	n9+n10+n13+n14=C
3---+ 4	11---+12	n1+n3+n5+n7=C	n9+n11+n13+n15=C
5-- - 6	13-- -14	n2+n4+n6+n8=C	n10+n12+n14+n16=C
7----- 8	15-----16	n3+n4+n7+n8=C	n11+n12+n15+n16=C
		n5+n6+n7+n8=C	n13+n14+n15+n16=C
2/d0	/d1	n1+n2+n3+n4=C	n9+n10+n11+n12=C
1----- 3	9-----11	n1+n3+n5+n7=C	n9+n11+n13+n15=C
2---+ 4	10---+12	n1+n2+n5+n6=C	n9+n10+n13+n14=C
5-- - 7	13-- -15	n3+n4+n7+n8=C	n11+n12+n15+n16=C
6----- 8	14-----16	n2+n4+n6+n8=C	n10+n12+n14+n16=C
		n5+n6+n7+n8=C	n13+n14+n15+n16=C
3/d0	/d1	n1+n2+n5+n6=C	n9+n10+n13+n14=C
1----- 2	9-----10	n1+n2+n3+n4=C	n9+n10+n11+n12=C
5---+ 6	13---+14	n1+n3+n5+n7=C	n9+n11+n13+n15=C
3-- - 4	11-- -12	n2+n4+n6+n8=C	n10+n12+n14+n16=C
7----- 8	15-----16	n5+n6+n7+n8=C	n13+n14+n15+n16=C
		n3+n4+n7+n8=C	n11+n12+n15+n16=C
4/d0	/d1	n1+n3+n5+n7=C	n9+n11+n13+n15=C
1----- 3	9-----11	n1+n2+n3+n4=C	n9+n10+n11+n12=C
5---+ 7	13---+15	n1+n2+n5+n6=C	n9+n10+n13+n14=C
2-- - 4	10-- -12	n3+n4+n7+n8=C	n11+n12+n15+n16=C
6----- 8	14-----16	n5+n6+n7+n8=C	n13+n14+n15+n16=C
		n2+n4+n6+n8=C	n10+n12+n14+n16=C
5/d0	/d1	n1+n2+n5+n6=C	n9+n10+n13+n14=C
1----- 5	9-----13	n1+n3+n5+n7=C	n9+n11+n13+n15=C
2---+ 6	10---+14	n1+n2+n3+n4=C	n9+n10+n11+n12=C
3-- - 7	11-- -15	n5+n6+n7+n8=C	n13+n14+n15+n16=C
4----- 8	12-----16	n2+n4+n6+n8=C	n10+n12+n14+n16=C
		n3+n4+n7+n8=C	n11+n12+n15+n16=C

6/d0	/d1	n1+n3+n5+n7=C	n9+n11+n13+n15=C
1---5	9---13	n1+n2+n5+n6=C	n9+n10+n13+n14=C
3---7	11---15	n1+n2+n3+n4=C	n9+n10+n11+n12=C
2-- -6	10-- -14	n5+n6+n7+n8=C	n13+n14+n15+n16=C
4----8	12----16	n3+n4+n7+n8=C	n11+n12+n15+n16=C
		n2+n4+n6+n8=C	n10+n12+n14+n16=C

See the 6 diagrams above. They look very similar to each other. Yes. They are made by reflection or rotation of a representative one. As they have the same meaning for our purpose, we don't have to have them all, only when we take a representative one out of these six. Then we know we have to prepare only four different diagrams, a sixth of 24.

Let's take the case of ECO2<sup>6</sup> next, and consider about the same problem.

1/	/N2i	2/	/N2i	3/	/N2i
1 17	000000 010000	1 17	000000 010000	1 33	000000 100000
2 18	000001 010001	2 18	000001 010001	9 41	001000 101000
3 19	000010 010010	9 25	001000 011000	17 49	010000 110000
4 20	000011 010011	10 26	001001 011001	25 57	011000 111000
5 21	000100 010100	3 19	000010 010010	2 34	000001 100001
6 22	000101 010101	4 20	000011 010011	10 42	001001 101001
7 23	000110 010110	11 27	001010 011010	18 50	010001 110001
8 24	000111 010111	12 28	001011 011011	26 58	011001 111001
9 25	001000 011000	5 21	000100 010100	3 35	000010 100010
10 26	001001 011001	6 22	000101 010101	11 43	001010 101010
11 27	001010 011010	13 29	001100 011100	19 51	010010 110010
12 28	001011 011011	14 30	001101 011101	27 59	011010 111010
13 29	001100 011100	7 23	000110 010110	4 36	000011 100011
14 30	001101 011101	8 24	000111 010111	12 44	001011 101011
15 31	001110 011110	15 31	001110 011110	20 52	010011 110011
16 32	001111 011111	16 32	001111 011111	28 60	011011 111011
33 49	100000 110000	33 49	100000 110000	5 37	000100 100100
34 50	100001 110001	34 50	100001 110001	13 45	001100 101100
35 51	100010 110010	41 57	101000 111000	21 53	010100 110100
36 52	100011 110011	42 58	101001 111001	29 61	011100 111100
37 53	100100 110100	35 51	100010 110010	6 38	000101 100101
38 54	100101 110101	36 52	100011 110011	14 46	001101 101101
39 55	100110 110110	43 59	101010 111010	22 54	010101 110101
40 56	100111 110111	44 60	101011 111011	30 62	011101 111101
41 57	101000 111000	37 53	100100 110100	7 39	000110 100110
42 58	101001 111001	38 54	100101 110101	15 47	001110 101110
43 59	101010 111010	45 61	101100 111100	23 55	010110 110110
44 60	101011 111011	46 62	101101 111101	31 63	011110 111110
45 61	101100 111100	39 55	100110 110110	8 40	000111 100111
46 62	101101 111101	40 56	100111 110111	16 48	001111 101111
47 63	101110 111110	47 63	101110 111110	24 56	010111 110111
48 64	101111 111111	48 64	101111 111111	32 64	011111 111111

The next numbers adjacent to the origin n1 must be n2, n3, n5, n9, n17 and n33. They will appear in so many different ways as 720, because 6P6=720.

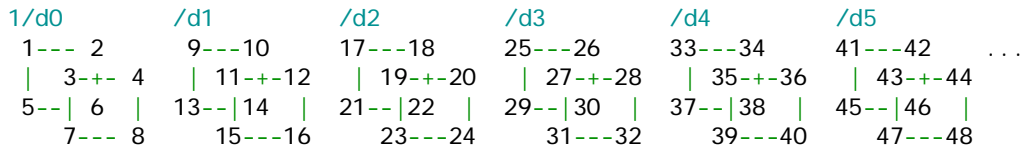
Permutation of {d0, d1, d2, d3, d4, d5} is also equal to the same count 720.

Therefore we can draw so many different Basic Diagrams as 720.

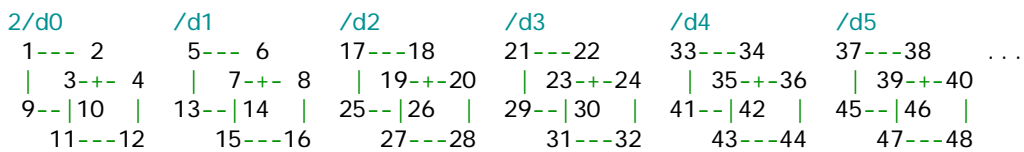
But it sounds too many for us to deal with.

As you guess we can remove some simple reflections and rotations. We have to prepare only 120 ones, a sixth of 720.

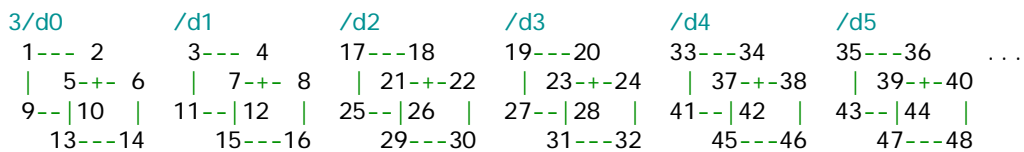
If you examine these 120 patterns more carefully, you can take the essential 20 diagrams for the definition stage. It is actually a reasonable count at last.



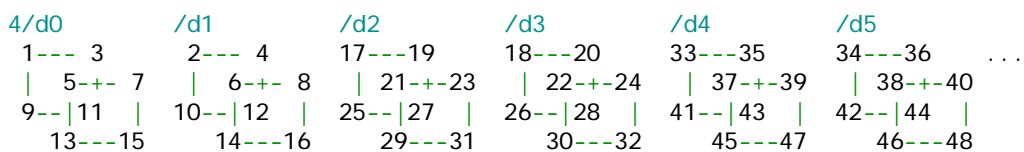
n1+n2+n3+n4=C	n9+n10+n11+n12=C	n17+n18+n19+n20=C	n25+n26+n27+n28=C
n1+n2+n5+n6=C	n9+n10+n13+n14=C	n17+n18+n21+n22=C	n25+n26+n29+n30=C
n1+n3+n5+n7=C	n9+n11+n13+n15=C	n17+n19+n21+n23=C	n25+n27+n29+n31=C
n2+n4+n6+n8=C	n10+n12+n14+n16=C	n18+n20+n22+n24=C	n26+n28+n30+n32=C
n3+n4+n7+n8=C	n11+n12+n15+n16=C	n19+n20+n23+n24=C	n27+n28+n31+n32=C
n5+n6+n7+n8=C	n13+n14+n15+n16=C	n21+n22+n23+n24=C	n29+n30+n31+n32=C



n1+n2+n3+n4=C	n5+n6+n7+n8=C	n17+n18+n19+n20=C	n21+n22+n23+n24=C
n1+n2+n9+n10=C	n5+n6+n13+n14=C	n17+n18+n25+n26=C	n21+n22+n29+n30=C
n1+n3+n9+n11=C	n5+n7+n13+n15=C	n17+n19+n25+n27=C	n21+n23+n29+n31=C
n2+n4+n10+n12=C	n6+n8+n14+n16=C	n18+n20+n26+n28=C	n22+n24+n30+n32=C
n3+n4+n11+n12=C	n7+n8+n15+n16=C	n19+n20+n27+n28=C	n23+n24+n31+n32=C
n9+n10+n11+n12=C	n13+n14+n15+n16=C	n25+n26+n27+n28=C	n29+n30+n31+n32=C



n1+n2+n5+n6=C	n3+n4+n7+n8=C	n17+n18+n21+n22=C	n19+n20+n23+n24=C
n1+n2+n9+n10=C	n3+n4+n11+n12=C	n17+n18+n25+n26=C	n19+n20+n27+n28=C
n1+n5+n9+n13=C	n3+n7+n11+n15=C	n17+n21+n25+n29=C	n19+n23+n27+n31=C
n2+n6+n10+n14=C	n4+n8+n12+n16=C	n18+n22+n26+n30=C	n20+n24+n28+n32=C
n5+n6+n13+n14=C	n7+n8+n15+n16=C	n21+n22+n29+n30=C	n23+n24+n31+n32=C
n9+n10+n13+n14=C	n11+n12+n15+n16=C	n25+n26+n29+n30=C	n27+n28+n31+n32=C



n1+n3+n5+n7=C	n2+n4+n6+n8=C	n17+n19+n21+n23=C	n18+n20+n22+n24=C
n1+n3+n9+n11=C	n2+n4+n10+n12=C	n17+n19+n25+n27=C	n18+n20+n26+n28=C
n1+n5+n9+n13=C	n2+n6+n10+n14=C	n17+n21+n25+n29=C	n18+n22+n26+n30=C
n3+n7+n11+n15=C	n4+n8+n12+n16=C	n19+n23+n27+n31=C	n20+n24+n28+n32=C
n5+n7+n13+n15=C	n6+n8+n14+n16=C	n21+n23+n29+n31=C	n22+n24+n30+n32=C
n9+n11+n13+n15=C	n10+n12+n14+n16=C	n25+n27+n29+n31=C	n26+n28+n30+n32=C

.....

How about the case of ECO3<sup>4</sup>? How many pictures can we draw? How many diagrams do we have to prepare for definition?

Calculation  $4P_4=4 \times 3 \times 2 \times 1=24$  is the answer. We only have to make 4 diagrams to prepare for definition, a sixth of 24.

1/

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

2/

1	10	19	28	37	46	55	64	73
2	11	20	29	38	47	56	65	74
3	12	21	30	39	48	57	66	75
4	13	22	31	40	49	58	67	76
5	14	23	32	41	50	59	68	77
6	15	24	33	42	51	60	69	78
7	16	25	34	43	52	61	70	79
8	17	26	35	44	53	62	71	80
9	18	27	36	45	54	63	72	81

How about the case of ECO2<sup>8</sup>?  
 Won't you think a little of it by yourself? Let me give you this 'homework'.

\*\*\* Study of Extra-Cubic Object of Order 2<sup>8</sup> \*\*\*

Study #81: Basic Diagram for Magic Square of Order 16<sup>2</sup>

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240
241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256

Study #82: Basic Diagram for Magic Square of Order 32x(2<sup>3</sup>)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96
97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112
113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160
161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176
177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192

193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208  
209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224  
225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240  
241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256

(Written in English on MacOS X & Xcode 1.5; May 9, 2005; Kanji Setsuda)

---

\*\*\* E-Mail Address: <jag12001@nifty.ne.jp>