Chapter 4: Commentary Articles No.2: by Kanji Setsuda
"Various Arts and Tools for Studying Magic Squares"
Section 7-4: Compose All the Solutions of 3 Types of Magic Squares of Order 4 only by the 4-th Increment Number System

0. I would like to report here about my new challenge and its result: How I tried to compose all the solutions of three types of MS44 only by the Positional Number System of the Base 4, namely, the 4-th Increment Number System.

I was often asked if it is possible for us to make all the solutions of typical MS44 only by the 4-th Increment Number System, not by our 'Binary System', but by the more natural and ordinary method as the 'N-th Increment System' for order N. It was my habit to answer "No, impossible!" to that kind of questions, to my regret. I am afraid I was a little too cold. I myself cannot not help feeling it seems natural to ask how it is possible, I must confess.

But, why can't we produce all only by the 4-th Increment System? Because we used to make such the 'Latin Units' on the way, only by using (0, 1, 2 and 3) to compose every rows, columns and diagonals for their contents, as I explained before. It has been the common practice for us to obey for a long time.

Why do we always obey that kind of conventional custom? I cannot but say it has been one of the oldest 'promises', whenever we should make magic things.

Why don't we take any new method now instead of such an old custom? Let's think what we can do, without using what they call 'Greco-Latinian Method'.

1. Let's use the 4-th Increment System here, but let's not stop using our "New Euler's Method" to make our objects. We must know about the inner structure of object by applying the two-way method of decomposition and composition by the 4-th Increment System. Watch the next conceptual diagrams, please.

```
** [1] Decomposition of Self-Complementary MS44 by D4i: **
      Classical/   Mathematical/ -> EUnit: H/         L/
   .--.--.--.--.   .--.--.--.--.    .-.-.-.-.  .-.-.-.-.
   | 1|15|14| 4|   | 0|14|13| 3|    |0|3|3|0|  |0|2|1|3|
   |--+--+--+--|   |--+--+--+--|    |-+-+-+-|  |-+-+-+-|
   |12| 6| 7| 9|   |11| 5| 6| 8|    |2|1|1|2|  |3|1|2|0|
   |--+--+--+--|   |--+--+--+--|    |-+-+-+-|  |-+-+-+-|
   | 8|10|11| 5|   | 7| 9|10| 4|    |1|2|2|1|  |3|1|2|0|
   |--+--+--+--|   |--+--+--+--|    |-+-+-+-|  |-+-+-+-|
   |13| 3| 2|16|   |12| 2| 1|15|    |3|0|0|3|  |0|2|1|3|
   '--'--'--'--'   '--'--'--'--'    '-'-'-'-'  '-'-'-'-'
  [Classical - 1 = Mathematical; Divide every number by 4 and
    put the Integer Part of Quotient to High Unit, the Remainder to Low]

** [2] Composition of Self-Complementary MS44 by D4i: **
   EUnit: H/         L/ -> Mathematical/      Classical/
   .-.-.-.-.  .-.-.-.-.    .--.--.--.--.   .--.--.--.--.
   |0|3|3|0|  |0|2|1|3|    | 0|14|13| 3|   | 1|15|14| 4|
   |-+-+-+-|  |-+-+-+-|    |--+--+--+--|   |--+--+--+--|
   |2|1|1|2|  |3|1|2|0|    |11| 5| 6| 8|   |12| 6| 7| 9|
   |-+-+-+-|  |-+-+-+-|    |--+--+--+--|   |--+--+--+--|
   |1|2|2|1|  |3|1|2|0|    | 7| 9|10| 4|   | 8|10|11| 5|
   |-+-+-+-|  |-+-+-+-|    |--+--+--+--|   |--+--+--+--|
   |3|0|0|3|  |0|2|1|3|    |12| 2| 1|15|   |13| 3| 2|16|
   '-'-'-'-'  '-'-'-'-'    '--'--'--'--'   '--'--'--'--'
  [(Every number in High Unit) x 4 + Corresponding Number in Low
         = Mathematical Expression/; + 1 = Classical One/]
```

As you see, these two sets of diagrams show the symmetric processes between the decomposition and composition of our object. The former one uses division by 4 and subtraction, and the latter multiplication by 4 and addition.

A pair of two little squares between them are called “Euler Units”. They are the results of decomposition and the components for composition at the same time.

Let’s take these **Euler Units** again, but let’s not call them “Latin Units” now.

Each unit consists of similar contents: any of 4 elements (0, 1, 2 and 3) comes just 4 times there as often as the others. We can also find no definite difference between the two units for high and low. That is one of the fatal characteristics of decomposed patterns of the base 4. It is also because the object squares are made after the basic promise that we must use any number from 0 to 15(math) strictly once and not un-use any one of those 16 numbers at all.

Now let’s deal with each unit equally, whether it is in high position or low, as in the same category on the same ground. It is one of the most important points of our New Euler’s Method.

2. The next problem is how to produce such the Euler Units one by one, pick up any two of the products to combine and compose our objective MS44 in the end.

Now we need to choose our practical type of MS44, and study the actual units decomposed by the 4-th Increment System.

Our success depends on what we can find among them, as you might guess.

It is only possible for us to observe them very well. Nothing else. By our earnest observation we have to find the new theoretical method itself out of the facts.

3. Let’s take the **Self-complementary type of MS44** at first.

```
** 'Self-complementary' Magic Squares 4x4 made by Ordinary Method **
 ** List of Standard Solutions with the Decomposed Units by N4i: **
          1#  /D4i                 2#  /D4i                 3#  /D4i
  1 15 14  4  0330 0213    1 15 14  4  0330 0213    1 15 12  6  0321 0231
 12  6  7  9  2112 3120    8 10 11  5  1221 3120   14  4  7  9  3012 1320
  8 10 11  5  1221 3120   12  6  7  9  2112 3120    8 10 13  3  1230 3102
 13  3  2 16  3003 0213   13  3  2 16  3003 0213   11  5  2 16  2103 2013
          4#  /D4i                 5#  /D4i                 6#  /D4i
  1 15 12  6  0321 0231    1 14 15  4  0330 0123    1 14 15  4  0330 0123
  8 10 13  3  1230 3102   12  7  6  9  2112 3210    8 11 10  5  1221 3210
 14  4  7  9  3012 1320    8 11 10  5  1221 3210   12  7  6  9  2112 3210
 11  5  2 16  2103 2013   13  2  3 16  3003 0123   13  2  3 16  3003 0123
          7#  /D4i                 8#  /D4i                 9#  /D4i
  1 14 12  7  0321 0132    1 14 12  7  0321 0132    1 12 15  6  0231 0321
 15  4  6  9  3012 2310    8 11 13  2  1230 3201   14  7  4  9  3102 1230
  8 11 13  2  1230 3201   15  4  6  9  3012 2310    8 13 10  3  1320 3012
 10  5  3 16  2103 1023   10  5  3 16  2103 1023   11  2  5 16  2013 2103
         10#  /D4i                11#  /D4i                12#  /D4i
  1 12 15  6  0231 0321    1 12 14  7  0231 0312    1 12 14  7  0231 0312
  8 13 10  3  1320 3012   15  6  4  9  3102 2130    8 13 11  2  1320 3021
 14  7  4  9  3102 1230    8 13 11  2  1320 3021   15  6  4  9  3102 2130
 11  2  5 16  2013 2103   10  3  5 16  2013 1203   10  3  5 16  2013 1203
         13#  /D4i                15#  /D4i                17#  /D4i
  2 16 13  3  0330 1302    2 16 11  5  0321 1320    2 13 16  3  0330 1032
 11  5  8 10  2112 2031   13  3  8 10  3012 0231   11  8  5 10  2112 2301
  7  9 12  6  1221 2031    7  9 14  4  1230 2013    7 12  9  6  1221 2301
 14  4  1 15  3003 1302   12  6  1 15  2103 3102   14  1  4 15  3003 1032
```

```
         19#  /D4i                21#  /D4i                23#  /D4i
 2 13 11  8  0321 1023    2 11 16  5  0231 1230    2 11 13  8  0231 1203
16  3  5 10  3012 3201   13  8  3 10  3102 0321   16  5  3 10  3102 3021
 7 12 14  1  1230 2310    7 14  9  4  1320 2103    7 14 12  1  1320 2130
 9  6  4 15  2103 0132   12  1  6 15  2013 3012    9  4  6 15  2013 0312
         25#  /D4i                27#  /D4i                29#  /D4i
 3 16 10  5  0321 2310    3 13 10  8  0321 2013    3 10 16  5  0231 2130
13  2  8 11  3012 0132   16  2  5 11  3012 3102   13  8  2 11  3102 0312
 6  9 15  4  1230 1023    6 12 15  1  1230 1320    6 15  9  4  1320 1203
12  7  1 14  2103 3201    9  7  4 14  2103 0231   12  1  7 14  2013 3021
         31#  /D4i                33#  /D4i                35#  /D4i
 3 10 13  8  0231 2103    4 15  9  6  0321 3201    4 14  9  7  0321 3102
16  5  2 11  3102 3012   14  1  7 12  3012 1023   15  1  6 12  3012 2013
 6 15 12  1  1320 1230    5 10 16  3  1230 0132    5 11 16  2  1230 0231
 9  4  7 14  2013 0321   11  8  2 13  2103 2310   10  8  3 13  2103 1320
         37#  /D4i                39#  /D4i                41#  /D4i
 4  9 15  6  0231 3021    4  9 14  7  0231 3012    5 11 10  8  1221 0213
14  7  1 12  3102 1203   15  6  1 12  3102 2103   16  2  3 13  3003 3120
 5 16 10  3  1320 0312    5 16 11  2  1320 0321    4 14 15  1  0330 3120
11  2  8 13  2013 2130   10  3  8 13  2013 1230    9  7  6 12  2112 0213
         43#  /D4i                45#  /D4i                47#  /D4i
 5 10 11  8  1221 0123    6 12  9  7  1221 1302    6  9 12  7  1221 1032
16  3  2 13  3003 3210   15  1  4 14  3003 2031   15  4  1 14  3003 2301
 4 15 14  1  0330 3210    3 13 16  2  0330 2031    3 16 13  2  0330 2301
 9  6  7 12  2112 0123   10  8  5 11  2112 1302   10  5  8 11  2112 1032
[Total Count of Standard Solutions = 48]
```

Will you watch them carefully and try to find something in the list, please? What do you find in each Euler Unit above equally in high position and low? The following list explains what I have found and analyzed about them:

(1) Any two elements in symmetrical positions make the 'Complementary Pair' of 3, such as (0, 3), (1, 2), (2, 1) and (3,0). Let me express it in the definition form:

```
n1+n16=n2+n15=n3+n14=n4+n13=n5+n12=n6+n11=n7+n10=n8+n9=n9+n8=...=3;
```

I found no other pairs existing between the symmetrical positions. This is the same kind of nature as the one our objective MS44SC has.

(2) Every 4 elements in any row and any column add up to the value 6. Every 4 numbers in the two primary diagonals also add up to 6.

```
.--.--.--.--.     * Rows and Columns: C=6; *
|n1|n2|n3|n4|       n1+ n2+ n3+ n4 = C ... rw1
|--+--+--+--|       n5+ n6+ n7+ n8 = C ... rw2
|n5|n6|n7|n8|       n9+n10+n11+n12 = C ... rw3
|--+--+--+--|      n13+n14+n15+n16 = C ... rw4
|n9|10|11|12|       n1+ n5+ n9+n13 = C ... cl1
|--+--+--+--|       n2+ n6+n10+n14 = C ... cl2
|13|14|15|16|       n3+ n7+n11+n15 = C ... cl3
'--'--'--'--'       n4+ n8+n12+n16 = C ... cl4
* Constant Sums of Two Primary Diagonals: C=6; *
  n1+ n6+n11+n16 = C  ... pd1;  n4+ n7+n10+n16 = C  ... pd2
```

No exceptions. This is the same kind of nature as the one our MS44SC has.

(3) In each Euler Unit any number of (0, 1, 2 and 3) comes just 4 times, as often as the others do. No exceptions are found. You can find no numbers coming 3 times nor 5 there at all, for instance.

4. We can use these 3 rules now when we produce Euler Units, don't you think?
And we can use the product Units when we pick up any two of them to combine and compose our MS44SC by multiplications and additions, can't we?
Let's try to do this now under the new rules and definitions above, shall we?

The next list shows my recent 'C' program, but only the core part.

```
//** 'New-Euler Type' of Self-complementary Magic Squares 4x4 **
//** Composed by "New Euler's Method" with /D4i **
//** 'NEulerMS44SCD4i.c' built by Kanji Setsuda **
//**  on Dec.13, 2011 and Jan.22, 2012  **
//**  with MacOSX 10.7.2 & Xcode 4.2.1  **
//
//** Basic Form for Self-complementary Magic Square of Order 4 **
//     .--.--.--.--.     * Rows and Columns: C=34; *
//    4|n1|n2|n3|n4| 1    n1+ n2+ n3+ n4 = C ... rw1
//     |--+--+--+--|      n5+ n6+ n7+ n8 = C ... rw2
//    8|n5|n6|n7|n8| 5    n9+n10+n11+n12 = C ... rw3
//     |--+--+--+--|     n13+n14+n15+n16 = C ... rw4
//   12|n9|10|11|12| 9    n1+ n5+ n9+n13 = C ... cl1
//     |--+--+--+--|      n2+ n6+n10+n14 = C ... cl2
//   16|13|14|15|16|13    n3+ n7+n11+n15 = C ... cl3
//     '--'--'--'--'      n4+ n8+n12+n16 = C ... cl4
//* Constant Sums of Self-complementary Pairs *
// n1+n16 = n2+n15 = n3+n14 = n4+n13 = n5+n12
// = n6+n11 = n7+n10 = n8+n9 = n9+n8 = n10+n7 = ... =17;
//
#include <stdio.h>
//
short int cnt, ecnt, cnt2, cnt3;
short luc[13];
short u1,u2;
short nm[17], uflg[17];
short anm[13][30];
short teun[49][16];
short mtc[49][49];
//
void estp01(void), estp02(void), estp03(void), estp04(void), estp05(void);
void estp06(void), estp07(void), estp08(void), estp09(void), estp10(void);
void estp11(void), estp12(void), estp13(void), estp14(void), estp15(void);
void estp16(void);
void eansrecord(void);
void prlunt(short x);
void tblmtc(void);
//
void cmbcmp(void);
void recprans(void), prans(short x);
//
//* Main Program *
int main(void){
 short n;
  printf("\n");
  printf("*** Make Self-complementary Magic Squares 4x4 by /D4i ***\n");
  for(n=0;n<17;n++){nm[n]=0;}
  for(n=0;n<4;n++){uflg[n]=0;}
  cnt=0; cnt3=0;
  printf(" ** Make Euler Units by /D4i **\n");
  estp01();                       // Make the Euler Units
  if(cnt3>0){prlunt(cnt3);}
  ecnt=cnt; printf(" [Total Count of Euler Units = %d]\n",ecnt);
  tblmtc();
  printf("\n");
  printf("** List of Self-complementary Magic Squares 4x4 made by /D4i **\n");
```

```
  cnt=0; cnt3=0;
  cmbcmp();                       // Combine, Compose and Print
  if(cnt3>0){prans(cnt3);}        // Print the Rest One
  printf(" [Solution Counts = %d]  [OK!]\n",cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on Jan.22, 2012 with MacOSX 10.7.2 & Xcode 4.2.1 **\n");
  printf("\n");
 return 0;
}
//
// Make the Euler Units
// Level #1
// Set n1 & n16
void estp01(){
 short i,j;                                  //Define (i,j) for the Complementary Pair
  for(i=0;i<4;i++){j=3-i;                    // i+j==3
    if((uflg[i]<4)&&(uflg[j]<4)){            //Check if every numbers can come 4 times
      nm[1]=i; nm[16]=j;
      uflg[i]++; uflg[j]++;                  //Set the UsedFlags[]
      estp02();
      uflg[j]--; uflg[i]--;}                 //Reset the UsedFlags[]

  }
}
// Set n2 & n15
void estp02(){
 short i,j;
  for(i=0;i<4;i++){j=3-i;
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[2]=i; nm[15]=j;
      uflg[i]++; uflg[j]++;
      estp03();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n4 & n13
void estp03(){
 short i,j;
  for(i=0;i<4;i++){j=3-i;
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[4]=i; nm[13]=j;
      uflg[i]++; uflg[j]++;
      estp04();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n3=6-n1-n2-n4 & n14=6-n16-n15-n13
void estp04(){
 short i,j;
  i=6-nm[1]-nm[2]-nm[4];
  j=6-nm[16]-nm[15]-nm[13];                  //Check the Constant Sums of 2 Rows
  if((0<=i)&&(i<4)&&(i+j==3)){               //Check if n3 and n14 make Complementary Pair
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[3]=i; nm[14]=j;
      uflg[i]++; uflg[j]++;
      estp05();
      uflg[j]--; uflg[i]--;}
  }
}
// Level #2
// Set n5 & n12
void estp05(){
 short i,j;
  for(i=0;i<4;i++){j=3-i;
```

```
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[5]=i; nm[12]=j;
      uflg[i]++; uflg[j]++;
      estp06();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n9=6-n1-n5-n13 & n8=6-n16-n12-n4
void estp06(){
 short i,j;
  i=6-nm[1]-nm[5]-nm[13];
  j=6-nm[16]-nm[12]-nm[4];
  if((0<=i)&&(i<4)&&(i+j==3)){
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[9]=i; nm[8]=j;
      uflg[i]++; uflg[j]++;
      estp07();
      uflg[j]--; uflg[i]--;}
  }
}
// Level #3:
// Set n6 & n11
void estp07(){
 short i,j;
  for(i=0;i<4;i++){j=3-i;
    if((uflg[i]<4)&&(uflg[j]<4)){cnt2=0;
      nm[6]=i; nm[11]=j;
      uflg[i]++; uflg[j]++;
      estp08();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n7=6-n5-n6-n8 & n10=6-n2-n6-n14
void estp08(){
 short i,j;
  i=6-nm[5]-nm[6]-nm[8];
  j=6-nm[2]-nm[6]-nm[14];
  if((0<=i)&&(i<4)&&(i+j==3)){
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[7]=i; nm[10]=j;
      uflg[i]++; uflg[j]++;
      estp09();
      uflg[j]--; uflg[i]--;}
  }
}
// Check n3+n7+n11+n15=34 & n9+n10+n11+n12=34
void estp09(){
 short i,j;
  i=nm[3]+nm[7]+nm[11]+nm[15];
  j=nm[9]+nm[10]+nm[11]+nm[12];
  if((i==6)&&(j==6)){eansrecord();}
}
//
// Record the Answers
void eansrecord(){
 short n;
  for(n=0;n<16;n++){teun[cnt][n]=nm[n+1];}
  cnt++; cnt2++;
  if(cnt2==1){luc[cnt3]=cnt; luc[cnt3+1]=0;
    for(n=0;n<16;n++){anm[cnt3][n]=nm[n+1]; anm[cnt3+1][n]=0;}
    cnt3++; if(cnt3==12){prlunt(cnt3); cnt3=0;}
  }
}
//
// Print the List of Euler Units
```

```
void prlunt(short x){
 short l,l4,m,n;
  for(m=0;m<x;m++){printf("%5d/",luc[m]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){printf("  ");
      for(n=0;n<4;n++){printf("%d",anm[m][l4+n]);}}
    printf("\n");
  }
}
//
// Check how well they match and Save the Data to the Table
void tblmtc(){
 short h,l,n,d;
  for(h=0;h<ecnt;h++){
    for(l=0;l<ecnt;l++){d=0;
      for(n=0;n<16;n++){if(teun[h][n]==teun[l][n]){d++;}}
      mtc[h][l]=d;}
  }
}
//
// Combine, Compose and Print
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<ecnt;u1++){                    //Choose a unit for hoigh position
    for(u2=0;u2<ecnt;u2++){cnt2=0;           //Choose a unit for low position
      if(mtc[u1][u2]==4){                    //Know the Matching Data from the Table
        for(n=1;n<17;n++){uflg[n]=0;}        //Reset the UsedFlag[]
        for(n=0;n<16;n++){
          d=teun[u1][n]*4+teun[u2][n]+1;     //Compose by multiplication and Addition
          nm[n+1]=d; uflg[d]++;
        }
        fc=0;
        for(n=1;n<17;n++){if(uflg[n]==0){fc++;}}
        if(fc==0){                           //Eliminate Duplication or Dropout
          if((nm[1]<nm[4])&&(nm[4]<nm[13])&&(nm[1]<nm[16])){
            recprans();}}                    //Select and print 'Standard Solutions'
      }
    }
  }
}
//
// Print the Solution List
void recprans(){
 short n;
  cnt++; cnt2++;
  if(cnt2==1){
    luc[cnt3*3]=cnt;
    for(n=1;n<17;n++){anm[cnt3][n]=nm[n]; anm[cnt3+1][n]=0;}
    luc[cnt3*3+1]=u1+1; luc[cnt3*3+2]=u2+1;
    cnt3++; if(cnt3==3){prans(cnt3); cnt3=0;}
  }
}
//
// Print each Answer
void prans(short x){
 short lu1,lu2;
 short l,l4,m,n;
  for(m=0;m<x;m++){
    printf("%5d/%5d/%12d#",luc[m*3+1],luc[m*3+2],luc[m*3]);
    if((m+1)<x){printf("  ");}}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){
      lu1=luc[m*3+1]-1; lu2=luc[m*3+2]-1;
```

```
      printf("  ");
      for(n=0;n<4;n++){printf("%d",teun[lu1][l4+n]);}
      printf("  ");
      for(n=0;n<4;n++){printf("%d",teun[lu2][l4+n]);}
      printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
      if((m+1)<x){printf("  ");}
    }
    printf("\n");
  }
}
//
```

All those programs above show how to produce the Euler Units, how to pick up 2 units of the products to combine, and how to compose our object MS44SC in the end. After that we must always test each result, find and eliminate any solution that has unnecessary duplication or drop-out of any number there.

The following two lists show the practical results of my experiment:

(1) List of the 48 Euler Units for our object type

(2) List of the 48 Standard Solutions of MS44SC made by our New Euler's Method

```
** Make Self-complementary Magic Squares 4x4 by New Euler's Method **
 ** Make Euler Units by /D4i **
   1/    2/    3/    4/    5/    6/    7/    8/    9/   10/   11/   12/
 0132  0132  0123  0231  0231  0213  0330  0330  0321  0321  0312  0312
 2310  3201  3210  1320  3102  3120  1221  2112  1230  3012  2130  3021
 3201  2310  3210  3102  1320  3120  2112  1221  3012  1230  3021  2130
 1023  1023  0123  2013  2013  0213  3003  3003  2103  2103  1203  1203
  13/   14/   15/   16/   17/   18/   19/   20/   21/   22/   23/   24/
 1032  1023  1023  1230  1230  1221  1221  1203  1203  1320  1320  1302
 2301  2310  3201  0321  2103  0330  3003  2130  3021  0231  2013  2031
 2301  3201  2310  2103  0321  3003  0330  3021  2130  2013  0231  2031
 1032  0132  0132  3012  3012  2112  2112  0312  0312  3102  3102  1302
  25/   26/   27/   28/   29/   30/   31/   32/   33/   34/   35/   36/
 2031  2013  2013  2130  2130  2112  2112  2103  2103  2310  2310  2301
 1302  1320  3102  0312  1203  0330  3003  1230  3012  0132  1023  1032
 1302  3102  1320  1203  0312  3003  0330  3012  1230  1023  0132  1032
 2031  0231  0231  3021  3021  1221  1221  0321  0321  3201  3201  2301
  37/   38/   39/   40/   41/   42/   43/   44/   45/   46/   47/   48/
 3021  3021  3012  3012  3003  3003  3120  3102  3102  3210  3201  3201
 0312  1203  0321  2103  1221  2112  0213  0231  2013  0123  0132  1023
 1203  0312  2103  0321  2112  1221  0213  2013  0231  0123  1023  0132
 2130  2130  1230  1230  0330  0330  3120  1320  1320  3210  2310  2310
 [Total Count of Euler Units = 48]

 ** List of Self-complementary MS44 made by New Euler's Method **
   4/   10/          1#     4/   12/          2#     4/   17/          3#
 0231  0321   1 12 15  6   0231  0312   1 12 14  7   0231  1230   2 11 16  5
 1320  3012   8 13 10  3   1320  3021   8 13 11  2   1320  2103   7 14  9  4
 3102  1230  14  7  4  9   3102  2130  15  6  4  9   3102  0321  13  8  3 10
 2013  2103  11  2  5 16   2013  1203  10  3  5 16   2013  3012  12  1  6 15
   4/   20/          4#     4/   29/          5#     4/   32/          6#
 0231  1203   2 11 13  8   0231  2130   3 10 16  5   0231  2103   3 10 13  8
 1320  2130   7 14 12  1   1320  1203   6 15  9  4   1320  1230   6 15 12  1
 3102  3021  16  5  3 10   3102  0312  13  8  2 11   3102  3012  16  5  2 11
 2013  0312   9  4  6 15   2013  3021  12  1  7 14   2013  0321   9  4  7 14
```

```
  4/   37/           7#      4/   39/           8#      5/    9/           9#
0231  3021   4  9 15  6    0231  3012   4  9 14  7    0231  0321   1 12 15  6
1320  0312   5 16 10  3    1320  0321   5 16 11  2    3102  1230  14  7  4  9
3102  1203  14  7  1 12    3102  2103  15  6  1 12    1320  3012   8 13 10  3
2013  2130  11  2  8 13    2013  1230  10  3  8 13    2013  2103  11  2  5 16

  5/   11/          10#      5/   16/          11#      5/   21/          12#
0231  0312   1 12 14  7    0231  1230   2 11 16  5    0231  1203   2 11 13  8
3102  2130  15  6  4  9    3102  0321  13  8  3 10    3102  3021  16  5  3 10
1320  3021   8 13 11  2    1320  2103   7 14  9  4    1320  2130   7 14 12  1
2013  1203  10  3  5 16    2013  3012  12  1  6 15    2013  0312   9  4  6 15

  5/   28/          13#      5/   33/          14#      5/   38/          15#
0231  2130   3 10 16  5    0231  2103   3 10 13  8    0231  3021   4  9 15  6
3102  0312  13  8  2 11    3102  3012  16  5  2 11    3102  1203  14  7  1 12
1320  1203   6 15  9  4    1320  1230   6 15 12  1    1320  0312   5 16 10  3
2013  3021  12  1  7 14    2013  0321   9  4  7 14    2013  2130  11  2  8 13

  5/   40/          16#      7/    3/          17#      7/    6/          18#
0231  3012   4  9 14  7    0330  0123   1 14 15  4    0330  0213   1 15 14  4
3102  2103  15  6  1 12    1221  3210   8 11 10  5    1221  3120   8 10 11  5
1320  0321   5 16 11  2    2112  3210  12  7  6  9    2112  3120  12  6  7  9
2013  1230  10  3  8 13    3003  0123  13  2  3 16    3003  0213  13  3  2 16

  8/    3/          21#      9/    2/          25#     10/    1/          33#
0330  0123   1 14 15  4    0321  0132   1 14 12  7    0321  0132   1 14 12  7
2112  3210  12  7  6  9    1230  3201   8 11 13  2    3012  2310  15  4  6  9
1221  3210   8 11 10  5    3012  2310  15  4  6  9    1230  3201   8 11 13  2
3003  0123  13  2  3 16    2103  1023  10  5  3 16    2103  1023  10  5  3 16

 18/    3/          41#     19/    3/          45#
1221  0123   5 10 11  8    1221  0123   5 10 11  8
0330  3210   4 15 14  1    3003  3210  16  3  2 13
3003  3210  16  3  2 13    0330  3210   4 15 14  1
2112  0123   9  6  7 12    2112  0123   9  6  7 12    [Solution Counts = 48]
```

Let me modify the list, with sorting the data and referring to the old solutions.

```
** Smart List of Self-complementary MS44 made by New Euler's Method **
    (Unit_Nmbr: High/ Low/; Solution_Nmbr: [New] Old# Sum_Checks\/||)
 8/H   6/L   [ 1]     1#\34/34|     7/H   6/L   [ 2]     2#\34/34|
0330  0213    1 15 14  4|34|34|    0330  0213    1 15 14  4|34|34|
2112  3120   12  6  7  9|34|34|    1221  3120    8 10 11  5|34|34|
1221  3120    8 10 11  5|34|34|    2112  3120   12  6  7  9|34|34|
3003  0213   13  3  2 16|34|34|    3003  0213   13  3  2 16|34|34|

10/H   4/L   [ 3]     3#\34/34|     9/H   5/L   [ 4]     4#\34/34|
0321  0231    1 15 12  6|34|34|    0321  0231    1 15 12  6|34|34|
3012  1320   14  4  7  9|34|34|    1230  3102    8 10 13  3|34|34|
1230  3102    8 10 13  3|34|34|    3012  1320   14  4  7  9|34|34|
2103  2013   11  5  2 16|34|34|    2103  2013   11  5  2 16|34|34|

 8/H   3/L   [ 5]     5#\34/34|     7/H   3/L   [ 6]     6#\34/34|
0330  0123    1 14 15  4|34|34|    0330  0123    1 14 15  4|34|34|
2112  3210   12  7  6  9|34|34|    1221  3210    8 11 10  5|34|34|
1221  3210    8 11 10  5|34|34|    2112  3210   12  7  6  9|34|34|
3003  0123   13  2  3 16|34|34|    3003  0123   13  2  3 16|34|34|

10/H   1/L   [ 7]     7#\34/34|     9/H   2/L   [ 8]     8#\34/34|
0321  0132    1 14 12  7|34|34|    0321  0132    1 14 12  7|34|34|
3012  2310   15  4  6  9|34|34|    1230  3201    8 11 13  2|34|34|
1230  3201    8 11 13  2|34|34|    3012  2310   15  4  6  9|34|34|
2103  1023   10  5  3 16|34|34|    2103  1023   10  5  3 16|34|34|
```

```
 5/H  9/L  [ 9]    9#\34/34|    4/H 10/L  [10]   10#\34/34|
0231 0321   1 12 15  6|34|34|  0231 0321   1 12 15  6|34|34|
3102 1230  14  7  4  9|34|34|  1320 3012   8 13 10  3|34|34|
1320 3012   8 13 10  3|34|34|  3102 1230  14  7  4  9|34|34|
2013 2103  11  2  5 16|34|34|  2013 2103  11  2  5 16|34|34|

 5/H 11/L  [11]   11#\34/34|    4/H 12/L  [12]   12#\34/34|
0231 0312   1 12 14  7|34|34|  0231 0312   1 12 14  7|34|34|
3102 2130  15  6  4  9|34|34|  1320 3021   8 13 11  2|34|34|
1320 3021   8 13 11  2|34|34|  3102 2130  15  6  4  9|34|34|
2013 1203  10  3  5 16|34|34|  2013 1203  10  3  5 16|34|34|

 8/H 24/L  [13]   13#\34/34|    7/H 24/L  [14]   14#\34/34|
0330 1302   2 16 13  3|34|34|  0330 1302   2 16 13  3|34|34|
2112 2031  11  5  8 10|34|34|  1221 2031   7  9 12  6|34|34|
1221 2031   7  9 12  6|34|34|  2112 2031  11  5  8 10|34|34|
3003 1302  14  4  1 15|34|34|  3003 1302  14  4  1 15|34|34|

10/H 22/L  [15]   15#\34/34|    9/H 23/L  [16]   16#\34/34|
0321 1320   2 16 11  5|34|34|  0321 1320   2 16 11  5|34|34|
3012 0231  13  3  8 10|34|34|  1230 2013   7  9 14  4|34|34|
1230 2013   7  9 14  4|34|34|  3012 0231  13  3  8 10|34|34|
2103 3102  12  6  1 15|34|34|  2103 3102  12  6  1 15|34|34|

 8/H 13/L  [17]   17#\34/34|    7/H 13/L  [18]   18#\34/34|
0330 1032   2 13 16  3|34|34|  0330 1032   2 13 16  3|34|34|
2112 2301  11  8  5 10|34|34|  1221 2301   7 12  9  6|34|34|
1221 2301   7 12  9  6|34|34|  2112 2301  11  8  5 10|34|34|
3003 1032  14  1  4 15|34|34|  3003 1032  14  1  4 15|34|34|

10/H 15/L  [19]   19#\34/34|    9/H 14/L  [20]   20#\34/34|
0321 1023   2 13 11  8|34|34|  0321 1023   2 13 11  8|34|34|
3012 3201  16  3  5 10|34|34|  1230 2310   7 12 14  1|34|34|
1230 2310   7 12 14  1|34|34|  3012 3201  16  3  5 10|34|34|
2103 0132   9  6  4 15|34|34|  2103 0132   9  6  4 15|34|34|

 5/H 16/L  [21]   21#\34/34|    4/H 17/L  [22]   22#\34/34|
0231 1230   2 11 16  5|34|34|  0231 1230   2 11 16  5|34|34|
3102 0321  13  8  3 10|34|34|  1320 2103   7 14  9  4|34|34|
1320 2103   7 14  9  4|34|34|  3102 0321  13  8  3 10|34|34|
2013 3012  12  1  6 15|34|34|  2013 3012  12  1  6 15|34|34|

 5/H 21/L  [23]   23#\34/34|    4/H 20/L  [24]   24#\34/34|
0231 1203   2 11 13  8|34|34|  0231 1203   2 11 13  8|34|34|
3102 3021  16  5  3 10|34|34|  1320 2130   7 14 12  1|34|34|
1320 2130   7 14 12  1|34|34|  3102 3021  16  5  3 10|34|34|
2013 0312   9  4  6 15|34|34|  2013 0312   9  4  6 15|34|34|

10/H 34/L  [25]   25#\34/34|   10/H 48/L  [33]   33#\34/34|
0321 2310   3 16 10  5|34|34|  0321 3201   4 15  9  6|34|34|
3012 0132  13  2  8 11|34|34|  3012 1023  14  1  7 12|34|34|
1230 1023   6  9 15  4|34|34|  1230 0132   5 10 16  3|34|34|
2103 3201  12  7  1 14|34|34|  2103 2310  11  8  2 13|34|34|

19/H  6/L  [41]   41#\34/34|   19/H 24/L  [45]   45#\34/34|
1221 0213   5 11 10  8|34|34|  1221 1302   6 12  9  7|34|34|
3003 3120  16  2  3 13|34|34|  3003 2031  15  1  4 14|34|34|
0330 3120   4 14 15  1|34|34|  0330 2031   3 13 16  2|34|34|
2112 0213   9  7  6 12|34|34|  2112 1302  10  8  5 11|34|34|
[Solution Counts(New/Old) = 48/48]  [OK!]

** Calculated and Listed by Kanji Setsuda
    on Jan.22, 2012 with MacOSX 10.7.2 & Xcode 4.2.1 **
```

5. They prove we have really succeeded in reconstructing all the solutions of Self-complementary MS44 by the 4-th Increment System. We could neither make any other type of solutions, nor miss any one of necessary solutions at all.

What does it mean by our success?

The three rules above for the Euler Units are very similar to the original rules of object MS44SC. This similarity should imply about the inner structure of this type of object. Our two-way method of decomposition and composition by the 4-th Increment System reveals this concordance between the Units and objects.

6. How about the next object, **'Composite and Complete' Pan-magic Squares** of Order 4? Can we also reconstruct all the solutions by the 4-th Increment Number System, as many as 48 with our New Euler's Method?

But, what do they look like? Let me show you the original list of definition.

```
** Basic Form and the Model Solution: **
 14 15 16 13 14 15 16 13 14 15
          .--.--.--.--.              .--.--.--.--.
  2  3  4|n1|n2|n3|n4| 1  2  3       | 1|15| 4|14|
          |--+--+--+--|              |--+--+--+--|
  6  7  8|n5|n6|n7|n8| 5  6  7       |12| 6| 9| 7|
          |--+--+--+--|              |--+--+--+--|
 10 11 12|n9|10|11|12| 9 10 11       |13| 3|16| 2|
          |--+--+--+--|              |--+--+--+--|
 14 15 16|13|14|15|16|13 14 15       | 8|10| 5|11|
          '--'--'--'--'              '--'--'--'--'
  2  3  4  1  2  3  4  1  2  3
** Constant Sums of Rows and Columns: C=34; **
    n1+n2+n3+n4=C     ...rw1;  | n1+n5+n9+n13=C    ...cl1;
** 'Composite' Conditions: C=34; **
    n1+n2+n5+n6=C     ...c1;   | n2+n3+n6+n7=C     ...c2;
    n3+n4+n7+n8=C     ...c3;   | n4+n1+n8+n5=C     ...c4;
    n5+n6+n9+n10=C    ...c5;   | n6+n7+n10+n11=C   ...c6;
    n7+n8+n11+n12=C   ...c7;   | n8+n5+n12+n9=C    ...c8;
    n9+n10+n13+n14=C  ...c9;   | n10+n11+n14+n15=C ...c10;
    n11+n12+n15+n16=C ...c11;  | n12+n9+n16+n13=C  ...c12;
    n13+n14+n1+n2=C   ...c13;  | n14+n15+n2+n3=C   ...c14;
    n15+n16+n3+n4=C   ...c15;  | n16+n13+n4+n1=C   ...c16;
** Constant Sums of Complete type of 'Complements': CC=17; **
    n1+n11=n2+n12=n3+n9=n4+n10=n5+n15=n6+n16=n7+n13=n8+n14=CC ...cc
** List-forming Inequality Conditions for the Standard Solutions: **
    n1<n4, n4<n13, n1<n16
```

This type is one of the variations of Pan-diagonal Magic Squares 4x4. They all have the same solution sets of 48 standard solutions through every types.

Let's reconstruct all those 48 solutions by our New Euler's Method, shall we?

What conditions do we have to presuppose for our new Euler Units, then?

How about modifying the above list as follows?

```
** Constant Sums of Rows and Columns: C=6; **
    n1+n2+n3+n4=C     ...rw1;  | n1+n5+n9+n13=C    ...cl1;
** 'Composite' Conditions: C=6; **
    n1+n2+n5+n6=C     ...c1;   | n2+n3+n6+n7=C     ...c2;
    n3+n4+n7+n8=C     ...c3;   | n4+n1+n8+n5=C     ...c4;
    n5+n6+n9+n10=C    ...c5;   | n6+n7+n10+n11=C   ...c6;
    n7+n8+n11+n12=C   ...c7;   | n8+n5+n12+n9=C    ...c8;
    n9+n10+n13+n14=C  ...c9;   | n10+n11+n14+n15=C ...c10;
    n11+n12+n15+n16=C ...c11;  | n12+n9+n16+n13=C  ...c12;
    n13+n14+n1+n2=C   ...c13;  | n14+n15+n2+n3=C   ...c14;
    n15+n16+n3+n4=C   ...c15;  | n16+n13+n4+n1=C   ...c16;
```

```
 ** Constant Sums of Complete type of 'Complements': CC=3; **
    n1+n11=n2+n12=n3+n9=n4+n10=n5+n15=n6+n16=n7+n13=n8+n14=CC ...cc
```

About the 'third rule', we may well accept it as it is before: Any number of (0, 1, 2 and 3) should come just 4 times in each unit, as often as the other numbers.

The new program for this object was written by modifying the former one:

```
//
// Make the Euler Units
// Level #1:
// Set n1 & n11
void estp01(){
 short i,j;                                  //Define the Complete type of Complements
  for(i=0;i<4;i++){j=PSM-i;                  // i+j=PSM(==3)
    if((uflg[i]<4)&&(uflg[j]<4)){            //Check the 'third rule' above
      nm[1]=i; nm[11]=j;                     // n1+n11==3
      uflg[i]++; uflg[j]++;
      estp02();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n2 & n12
void estp02(){
 short i,j;
  for(i=0;i<4;i++){j=PSM-i;
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[2]=i; nm[12]=j;
      uflg[i]++; uflg[j]++;
      estp03();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n3 & n9
void estp03(){
 short i,j;
  for(i=0;i<4;i++){j=PSM-i;
    if((uflg[i]<4)&&(uflg[j]<4)){cnt2=0;
      nm[3]=i; nm[9]=j;
      uflg[i]++; uflg[j]++;
      estp04();
      uflg[j]--; uflg[i]--;}
  }
}
// Set n4=LSM-n1-n2-n3 & n10=LSM-n11-n12-n9
void estp04(){
 short i,j;
  i=LSM-nm[1]-nm[2]-nm[3];                   // n1+n2+n3+n4=LSM(==6)
  j=LSM-nm[11]-nm[12]-nm[9];                 // n9+n10+n11+n12=LSM(==6)
  if((0<=i)&&(i<4)&&(i+j==PSM)){             //Check if n4+n10==PSM(==3)
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[4]=i; nm[10]=j;
      uflg[i]++; uflg[j]++;
      estp05();
      uflg[j]--; uflg[i]--;}
  }
}
// Level #2
// Set n5 & n15
void estp05(){
 short i,j;
  for(i=0;i<4;i++){j=PSM-i;
    if((uflg[i]<4)&&(uflg[j]<4)){if(nm[1]==1){cnt2=0;}
      nm[5]=i; nm[15]=j;
      uflg[i]++; uflg[j]++;
```

```
        estp06();
        uflg[j]--; uflg[i]--;}
    }
}
// Set n13=LSM-n1-n5-n9 & n7=LSM-n11-n15-n3
void estp06(){
 short i,j;
  i=LSM-nm[1]-nm[5]-nm[9];
  j=LSM-nm[11]-nm[15]-nm[3];
  if((0<=i)&&(i<4)&&(i+j==PSM)){
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[13]=i; nm[7]=j;
      uflg[i]++; uflg[j]++;
      estp07();
      uflg[j]--; uflg[i]--;}
  }
}
// Level #3:
// Set n6=LSM-n1-n2-n5 & n16=LSM-n11-n12-n15
void estp07(){
 short i,j;
  i=LSM-nm[1]-nm[2]-nm[5];
  j=LSM-nm[11]-nm[12]-nm[15];
  if((0<=i)&&(i<4)&&(i+j==PSM)){
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[6]=i; nm[16]=j;
      uflg[i]++; uflg[j]++;
      estp08();
      uflg[j]--; uflg[i]--;}
  }
}
// Check n2+n3+n6+n7=6 & n5+n6+n9+n10=6 & n6+n7+n10+n11=6
void estp08(){
 short s1,s2,s3;
  s1=nm[2]+nm[3]+nm[6]+nm[7];
  s2=nm[5]+nm[6]+nm[9]+nm[10];
  s3=nm[6]+nm[7]+nm[10]+nm[11];
  if((s1==LSM)&&(s2==LSM)&&(s3==LSM)){estp09();}
}
// Set n8=LSM-n3-n4-n7 & n14=LSM-n9-n10-n13
void estp09(){
 short i,j;
  i=LSM-nm[3]-nm[4]-nm[7];
  j=LSM-nm[9]-nm[10]-nm[13];
  if((0<=i)&&(i<4)&&(i+j==PSM)){
    if((uflg[i]<4)&&(uflg[j]<4)){
      nm[8]=i; nm[14]=j;
      uflg[i]++; uflg[j]++;
      estp10();
      uflg[j]--; uflg[i]--;}
  }
}
// Check n7+n8+n11+n12=6 & n10+n11+n14+n15=6
void estp10(){
 short s1,s2;
  s1=nm[7]+nm[8]+nm[11]+nm[12];
  s2=nm[10]+nm[11]+nm[14]+nm[15];
  if((s1==LSM)&&(s2==LSM)){eansrecord();}
}
//
// Record the Answers
void eansrecord(){
 short n;
  for(n=0;n<16;n++){teun[cnt][n]=nm[n+1];}
  cnt++; cnt2++;
```

```
  if(cnt2==1){luc[cnt3]=cnt; luc[cnt3+1]=0;
    for(n=0;n<16;n++){anm[cnt3][n]=nm[n+1]; anm[cnt3+1][n]=0;}
    cnt3++; if(cnt3==12){prlunt(cnt3); cnt3=0;}
  }
}
//
// Print the List of Euler Units
void prlunt(short x){
 short l,l4,m,n;
  for(m=0;m<x;m++){printf("%5d/",luc[m]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){printf("   ");
      for(n=0;n<4;n++){printf("%d",anm[m][l4+n]);}}
    printf("\n");
  }
}
//
// Check how well they match and Save the Data to the Table
void tblmtc(){
 short h,l,n,d;
  for(h=0;h<ecnt;h++){
    for(l=0;l<ecnt;l++){d=0;
      for(n=0;n<16;n++){if(teun[h][n]==teun[l][n]){d++;}}
      mtc[h][l]=d;}
  }
}
//
// Combine, Compose and Print
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<ecnt;u1++){
    for(u2=0;u2<ecnt;u2++){cnt2=0;
      if(mtc[u1][u2]==4){
        for(n=1;n<17;n++){uflg[n]=0;}
        for(n=0;n<16;n++){
          d=teun[u1][n]*4+teun[u2][n]+1;
          nm[n+1]=d; uflg[d]++;
        }
        fc=0;
        for(n=1;n<17;n++){if(uflg[n]==0){fc++;}}
        if(fc==0){
          if((nm[1]<nm[4])&&(nm[4]<nm[13])&&(nm[1]<nm[16])){
            recprans();}}
      }
    }
  }
}
//
// Record and Print the Solution List
void recprans(){
 short n;
  cnt++; cnt2++;
  if(cnt2==1){
    luc[cnt3*3]=cnt;
    for(n=1;n<17;n++){anm[cnt3][n]=nm[n]; anm[cnt3+1][n]=0;}
    luc[cnt3*3+1]=u1+1; luc[cnt3*3+2]=u2+1;
    cnt3++; if(cnt3==3){prans(cnt3); cnt3=0;}
  }
}
//
// Print each Answer
void prans(short x){
 short lu1,lu2;
 short l,l4,m,n;
```

```
  for(m=0;m<x;m++){
    printf("%5d/%5d/%12d#",luc[m*3+1],luc[m*3+2],luc[m*3]);
    if((m+1)<x){printf("   ");}}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){
      lu1=luc[m*3+1]-1; lu2=luc[m*3+2]-1;
      printf("   ");
      for(n=0;n<4;n++){printf("%d",teun[lu1][l4+n]);}
      printf("   ");
      for(n=0;n<4;n++){printf("%d",teun[lu2][l4+n]);}
      printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
      if((m+1)<x){printf("   ");}
    }
    printf("\n");
  }
}
//
```

7. The following two lists show the practical results of my experiment:
(1) List of the new 48 Euler Units for our second type
(2) List of the 48 Standard Solutions of CCPMS44 by our New Euler's Method

```
*** Make 'Composite & Complete' Pan-magic Squares 4x4 by /D4i ***
 ** Make the Euler Units by /D4i **
    1/    3/    4/    6/    7/    9/   11/   13/   14/   15/   16/   17/
 0123  0132  0213  0231  0303  0312  0321  1023  1032  1032  1203  1203
 2301  3201  1302  3102  1212  1203  2103  2310  2301  3210  0312  2130
 1032  0132  2031  0231  3030  2130  1230  1023  0123  0123  3021  3021
 3210  3201  3120  3102  2121  3021  3012  2310  3210  2301  2130  0312

   18/   19/   20/   21/   22/   23/   24/   25/   26/   28/   30/   32/
 1212  1212  1230  1230  1302  1302  1320  2013  2031  2103  2121  2130
 0303  3030  2103  3012  0213  2031  2013  1320  1302  0321  0303  1203
 2121  2121  0321  0321  3120  3120  1320  2013  0213  3012  1212  0312
 3030  0303  3012  2103  2031  0213  2013  1320  3120  1230  3030  3021

   34/   36/   37/   39/   41/   43/   44/   46/   47/
 2301  2310  3012  3021  3030  3102  3120  3201  3210
 0123  1023  0321  0312  1212  0231  0213  0132  0123
 3210  2310  2103  1203  0303  3102  1302  3201  2301
 1032  1023  1230  2130  2121  0231  2031  0132  1032
[Total Count of Euler Units = 48]

* Smart List of 'Composite & Complete' Pan-magic Squares 4x4 made by /D4i *
    (Used_Units: High/ Low/; Solution_Nmbr: [New] Ordinary# Sum_Checks:||\/|)
  9/H   5/L  [ 1]     1#|Rw|Cl\Pd/P2|   12/H   4/L  [ 2]     2#|Rw|Cl\Pd/P2|
 0312  0213   1 15  6 12|34|34\34/34|   0321  0213   1 15 10  8|34|34\34/34|
 1203  3120   8 10  3 13|34|34\34/34|   3012  1302  14  4  5 11|34|34\34/34|
 2130  2031  11  5 16  2|34|34\34/34|   1230  2031   7  9 16  2|34|34\34/34|
 3021  1302  14  4  9  7|34|34\34/34|   2103  3120  12  6  3 13|34|34\34/34|

 11/H   5/L  [ 3]     3#|Rw|Cl\Pd/P2|    9/H   2/L  [ 4]     4#|Rw|Cl\Pd/P2|
 0321  0213   1 15 10  8|34|34\34/34|   0312  0123   1 14  7 12|34|34\34/34|
 2103  3120  12  6  3 13|34|34\34/34|   1203  3210   8 11  2 13|34|34\34/34|
 1230  2031   7  9 16  2|34|34\34/34|   2130  1032  10  5 16  3|34|34\34/34|
 3012  1302  14  4  5 11|34|34\34/34|   3021  2301  15  4  9  6|34|34\34/34|

 12/H   1/L  [ 5]     5#|Rw|Cl\Pd/P2|   11/H   2/L  [ 6]     6#|Rw|Cl\Pd/P2|
 0321  0123   1 14 11  8|34|34\34/34|   0321  0123   1 14 11  8|34|34\34/34|
 3012  2301  15  4  5 10|34|34\34/34|   2103  3210  12  7  2 13|34|34\34/34|
 1230  1032   6  9 16  3|34|34\34/34|   1230  1032   6  9 16  3|34|34\34/34|
 2103  3210  12  7  2 13|34|34\34/34|   3012  2301  15  4  5 10|34|34\34/34|
```

```
 4/H  12/L  [ 7]     7#|Rw|Cl\Pd/P2|    6/H   8/L  [ 8]     8#|Rw|Cl\Pd/P2|
0213  0321   1 12  7 14|34|34\34/34|   0231  0303   1 12 13  8|34|34\34/34|
1302  3012   8 13  2 11|34|34\34/34|   3102  2121  15  6  3 10|34|34\34/34|
2031  1230  10  3 16  5|34|34\34/34|   0231  3030   4  9 16  5|34|34\34/34|
3120  2103  15  6  9  4|34|34\34/34|   3102  1212  14  7  2 11|34|34\34/34|

 6/H   7/L  [ 9]     9#|Rw|Cl\Pd/P2|    1/H  12/L  [10]    10#|Rw|Cl\Pd/P2|
0231  0303   1 12 13  8|34|34\34/34|   0123  0321   1  8 11 14|34|34\34/34|
3102  1212  14  7  2 11|34|34\34/34|   2301  3012  12 13  2  7|34|34\34/34|
0231  3030   4  9 16  5|34|34\34/34|   1032  1230   6  3 16  9|34|34\34/34|
3102  2121  15  6  3 10|34|34\34/34|   3210  2103  15 10  5  4|34|34\34/34|

 3/H   8/L  [11]    11#|Rw|Cl\Pd/P2|    3/H   7/L  [12]    12#|Rw|Cl\Pd/P2|
0132  0303   1  8 13 12|34|34\34/34|   0132  0303   1  8 13 12|34|34\34/34|
3201  2121  15 10  3  6|34|34\34/34|   3201  1212  14 11  2  7|34|34\34/34|
0132  3030   4  5 16  9|34|34\34/34|   0132  3030   4  5 16  9|34|34\34/34|
3201  1212  14 11  2  7|34|34\34/34|   3201  2121  15 10  3  6|34|34\34/34|

 9/H  23/L  [13]    13#|Rw|Cl\Pd/P2|   12/H  22/L  [14]    14#|Rw|Cl\Pd/P2|
0312  1302   2 16  5 11|34|34\34/34|   0321  1302   2 16  9  7|34|34\34/34|
1203  2031   7  9  4 14|34|34\34/34|   3012  0213  13  3  6 12|34|34\34/34|
2130  3120  12  6 15  1|34|34\34/34|   1230  3120   8 10 15  1|34|34\34/34|
3021  0213  13  3 10  8|34|34\34/34|   2103  2031  11  5  4 14|34|34\34/34|

11/H  23/L  [15]    15#|Rw|Cl\Pd/P2|    9/H  14/L  [16]    16#|Rw|Cl\Pd/P2|
0321  1302   2 16  9  7|34|34\34/34|   0312  1032   2 13  8 11|34|34\34/34|
2103  2031  11  5  4 14|34|34\34/34|   1203  2301   7 12  1 14|34|34\34/34|
1230  3120   8 10 15  1|34|34\34/34|   2130  0123   9  6 15  4|34|34\34/34|
3012  0213  13  3  6 12|34|34\34/34|   3021  3210  16  3 10  5|34|34\34/34|

12/H  15/L  [17]    17#|Rw|Cl\Pd/P2|   11/H  14/L  [18]    18#|Rw|Cl\Pd/P2|
0321  1032   2 13 12  7|34|34\34/34|   0321  1032   2 13 12  7|34|34\34/34|
3012  3210  16  3  6  9|34|34\34/34|   2103  2301  11  8  1 14|34|34\34/34|
1230  0123   5 10 15  4|34|34\34/34|   1230  0123   5 10 15  4|34|34\34/34|
2103  2301  11  8  1 14|34|34\34/34|   3012  3210  16  3  6  9|34|34\34/34|

 4/H  20/L  [19]    19#|Rw|Cl\Pd/P2|    6/H  19/L  [20]    20#|Rw|Cl\Pd/P2|
0213  1230   2 11  8 13|34|34\34/34|   0231  1212   2 11 14  7|34|34\34/34|
1302  2103   7 14  1 12|34|34\34/34|   3102  3030  16  5  4  9|34|34\34/34|
2031  0321   9  4 15  6|34|34\34/34|   0231  2121   3 10 15  6|34|34\34/34|
3120  3012  16  5 10  3|34|34\34/34|   3102  0303  13  8  1 12|34|34\34/34|

 6/H  18/L  [21]    21#|Rw|Cl\Pd/P2|    1/H  20/L  [22]    22#|Rw|Cl\Pd/P2|
0231  1212   2 11 14  7|34|34\34/34|   0123  1230   2  7 12 13|34|34\34/34|
3102  0303  13  8  1 12|34|34\34/34|   2301  2103  11 14  1  8|34|34\34/34|
0231  2121   3 10 15  6|34|34\34/34|   1032  0321   5  4 15 10|34|34\34/34|
3102  3030  16  5  4  9|34|34\34/34|   3210  3012  16  9  6  3|34|34\34/34|

 3/H  19/L  [23]    23#|Rw|Cl\Pd/P2|    3/H  18/L  [24]    24#|Rw|Cl\Pd/P2|
0132  1212   2  7 14 11|34|34\34/34|   0132  1212   2  7 14 11|34|34\34/34|
3201  3030  16  9  4  5|34|34\34/34|   3201  0303  13 12  1  8|34|34\34/34|
0132  2121   3  6 15 10|34|34\34/34|   0132  2121   3  6 15 10|34|34\34/34|
3201  0303  13 12  1  8|34|34\34/34|   3201  3030  16  9  4  5|34|34\34/34|

 9/H  35/L  [25]    25#|Rw|Cl\Pd/P2|    9/H  47/L  [35]    35#|Rw|Cl\Pd/P2|
0312  2301   3 16  5 10|34|34\34/34|   0312  3210   4 15  6  9|34|34\34/34|
1203  1032   6  9  4 15|34|34\34/34|   1203  0123   5 10  3 16|34|34\34/34|
2130  3210  12  7 14  1|34|34\34/34|   2130  2301  11  8 13  2|34|34\34/34|
3021  0123  13  2 11  8|34|34\34/34|   3021  1032  14  1 12  7|34|34\34/34|

22/H  12/L  [45]    45#|Rw|Cl\Pd/P2|   22/H  20/L  [47]    47#|Rw|Cl\Pd/P2|
1302  0321   5 16  3 10|34|34\34/34|   1302  1230   6 15  4  9|34|34\34/34|
0213  3012   4  9  6 15|34|34\34/34|   0213  2103   3 10  5 16|34|34\34/34|
3120  1230  14  7 12  1|34|34\34/34|   3120  0321  13  8 11  2|34|34\34/34|
2031  2103  11  2 13  8|34|34\34/34|   2031  3012  12  1 14  7|34|34\34/34|
[Solution Counts(New/Ordinary) = 48/48]  [OK!]

** Calculated and Listed by Kanji Setsuda
    on Jan.23, 2012 with MacOSX 10.7.2 & Xcode 4.2.1 **
```

We have succeeded again in reconstructing all the solutions of 'Composite & Complete' Pan-magic Squares of Order 4 by the 4-th Increment Number System. We could neither make any other type of solutions, nor miss any necessary one.

The new 'three rules' above for the Euler Units imply about the inner structure of this type of object. It is revealed by our two-way method of decomposition and composition by the 4-th Increment Number System.

8. How about the third case of 'Standard' Magic Squares of Order 4?

We know there are 880 standard solutions for that type.

Can we really reconstruct all those solutions as many as 880 by our New Euler's Method and the 4-th Increment Number System?

Watch the next two lists of definitions I prepared: The first one shows the concept of our original type, while the similar second defines the conditions of their Euler Units, which I put here experimentally for a while.

```
** (1) Definitions for Standard Magic Squares of Order 4 **
  .--.--.--.--. * Basic Form and Equations: C=34 *
  | 1| 2| 3| 4|  n1+n2+n3+n4=C       ...rw1;
  |--+--+--+--|  n5+n6+n7+n8=C       ...rw2;
  | 5| 6| 7| 8|  n9+n10+n11+n12=C    ...rw3;
  |--+--+--+--|  n13+n14+n15+n16=C   ...rw4;
  | 9|10|11|12|  n1+n5+n9+n13=C      ...cl1;
  |--+--+--+--|  n2+n6+n10+n14=C     ...cl2;
  |13|14|15|16|  n3+n7+n11+n15=C     ...cl3;
  '--'--'--'--'  n4+n8+n12+n16=C     ...cl4;
   n1+n6+n11+n16=C ...Pd1;  n4+n7+n10+n13=C ...Pd2;
 ** List-forming Inequality Conditions **
  n1<n4; n1<n13; n1<n16; and n4<n13;

** (2) Definitions of Euler Units for Standard MS44 **
  .--.--.--.--. * Basic Form and Equations: C=6; *
  | 1| 2| 3| 4|  n1+n2+n3+n4=C       ...rw1;
  |--+--+--+--|  n5+n6+n7+n8=C       ...rw2;
  | 5| 6| 7| 8|  n9+n10+n11+n12=C    ...rw3;
  |--+--+--+--|  n13+n14+n15+n16=C   ...rw4;
  | 9|10|11|12|  n1+n5+n9+n13=C      ...cl1;
  |--+--+--+--|  n2+n6+n10+n14=C     ...cl2;
  |13|14|15|16|  n3+n7+n11+n15=C     ...cl3;
  '--'--'--'--'  n4+n8+n12+n16=C     ...cl4;
   n1+n6+n11+n16=C ...Pd1;  n4+n7+n10+n13=C ...Pd2;
```

That's all for the definition of our new Euler Units, but we must add the 'third rule' about the 4x4 numbers in each unit, the same rule with the one put before.

The following list contains the recent program of mine, only the core part.

```
//
//* Make the New Euler Units *
// Level #1:
// Set n1
void stp01(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[1]=i;
      uflg[i]++; stp02(); uflg[i]--;}
  }
}
// Set n2
void stp02(){
 short i;
  for(i=3;i>=0;i--){
    if(uflg[i]<4){nm[2]=i;
```

```
      uflg[i]++; stp03(); uflg[i]--;}
  }
}
// Set n4
void stp03(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[4]=i; cnt2=0;
      uflg[i]++; stp04(); uflg[i]--;}
  }
}
// Set n3
void stp04(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[3]=i;
      sm=nm[1]+nm[2]+nm[3]+nm[4];
      if(sm==LSM){uflg[i]++; stp05(); uflg[i]--;}}
  }
}
// Set n5
void stp05(){
 short i;
  for(i=3;i>=0;i--){
    if(uflg[i]<4){nm[5]=i;
      uflg[i]++; stp06(); uflg[i]--;}
  }
}
// Set n13
void stp06(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[13]=i;
      uflg[i]++; stp07(); uflg[i]--;}
  }
}
// Set n9
void stp07(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[9]=i;
      sm=nm[1]+nm[5]+nm[9]+nm[13];
      if(sm==LSM){uflg[i]++; stp08(); uflg[i]--;}}
  }
}
// Level #2:
// Set n6
void stp08(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[6]=i;
      uflg[i]++; stp09(); uflg[i]--;}
  }
}
// Set n7
void stp09(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[7]=i;
      uflg[i]++; stp10(); uflg[i]--;}
  }
}
// Set n10
void stp10(){
 short i,sm;
```

```
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[10]=i;
      sm=nm[4]+nm[7]+nm[10]+nm[13];
      if(sm==LSM){uflg[i]++; stp11(); uflg[i]--;}}
  }
}
// Set n8
void stp11(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[8]=i;
      sm=nm[5]+nm[6]+nm[7]+nm[8];
      if(sm==LSM){uflg[i]++; stp12(); uflg[i]--;}}
  }
}
// Set n14
void stp12(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[14]=i;
      sm=nm[2]+nm[6]+nm[10]+nm[14];
      if(sm==LSM){uflg[i]++; stp13(); uflg[i]--;}}
  }
}
// Level #3:
// Set n11
void stp13(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[11]=i;
      uflg[i]++; stp14(); uflg[i]--;}
  }
}
// Set n12
void stp14(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[12]=i;
      sm=nm[9]+nm[10]+nm[11]+nm[12];
      if(sm==LSM){uflg[i]++; stp15(); uflg[i]--;}}
  }
}
// Set n15
void stp15(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[15]=i;
      sm=nm[3]+nm[7]+nm[11]+nm[15];
      if(sm==LSM){uflg[i]++; stp16(); uflg[i]--;}}
  }
}
// Set n16
void stp16(){
 short i,sm1,sm2,sm3;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[16]=i;
      sm1=nm[13]+nm[14]+nm[15]+nm[16];
      sm2=nm[4]+nm[8]+nm[12]+nm[16];
      sm3=nm[1]+nm[6]+nm[11]+nm[16];
      if((sm1==LSM)&&(sm2==LSM)&&(sm3==LSM)){
        uflg[i]++; ansrecord(); uflg[i]--;}}
  }
}
//
// Record the Answers
```

```
void ansrecord(){
 short n;
  for(n=0;n<16;n++){tlu[cnt][n]=nm[n+1];}
  cnt++; cnt2++;
  if(cnt2==1){luc[cnt3]=cnt; luc[cnt3+1]=0;
    for(n=0;n<16;n++){anm[cnt3][n]=nm[n+1]; anm[cnt3+1][n]=0;}
    cnt3++; if(cnt3==12){pr12lun(cnt3); cnt3=0;}
  }
}
//
// Print Every 12 Euler Units
void pr12lun(short x){
 short l,l4,m,n;
  for(m=0;m<x;m++){printf("%5d/",luc[m]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){printf("  ");
      for(n=0;n<4;n++){printf("%d",anm[m][l4+n]);}}
    printf("\n");
  }
  printf("\n");
}
//
// Check how well they match and Save the Data to the Table
void tblmtc(){
 short h,l,n,d;
  for(h=0;h<cnt;h++){
    for(l=0;l<cnt;l++){d=0;
      for(n=1;n<17;n++){if(tlu[h][n]==tlu[l][n]){d++;}}
      mtc[h][l]=d;}
  }
}
//
// Combine, Compose and Print
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<(lcnt/2);u1++){cnt2=0;
    for(u2=0;u2<lcnt;u2++){
      if((2<mtc[u1][u2])&&(mtc[u1][u2]<5)){
        for(n=1;n<17;n++){uflg[n]=0;}
        for(n=0;n<16;n++){
          d=tlu[u1][n]*4+tlu[u2][n]+1;
          nm[n+1]=d; uflg[d]++;
        }
        fc=0;
        for(n=1;n<17;n++){if(uflg[n]==0){fc++;}}
        if(fc==0){
          if((nm[1]<nm[4])&&(nm[4]<nm[13])&&(nm[1]<nm[16])){
            recprans();}}
      }
    }
  }
}
//
// Record and Print the Solution List
void recprans(){
 short n;
  cnt++; cnt2++;
  if(cnt2==1){
    luc[cnt3*3]=cnt;
    for(n=1;n<17;n++){anm[cnt3][n]=nm[n]; anm[cnt3+1][n]=0;}
    luc[cnt3*3+1]=u1+1; luc[cnt3*3+2]=u2+1;
    cnt3++; if(cnt3==3){pr3ans(); cnt3=0;}
  }
}
```

```
//
// Print 3 Answers
void pr3ans(){
 short lu1,lu2;
 short l,l4,m,n;
  printf("%5d/%5d/%12d#%7d/%5d/%12d#%7d/%5d/%12d#\n",
         luc[1],luc[2],luc[0],luc[4],luc[5],luc[3],luc[7],luc[8],luc[6]);
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<3;m++){
      lu1=luc[m*3+1]-1; lu2=luc[m*3+2]-1;
      printf("  ");
      for(n=0;n<4;n++){printf("%d",tlu[lu1][l4+n]);}
      printf("  ");
      for(n=0;n<4;n++){printf("%d",tlu[lu2][l4+n]);}
      printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
      if(m<2){printf("  ");}
    }
    printf("\n");
  }
}
//
// Print the Rest One
void pr1ans(){
 short lu1,lu2;
 short l,l4,n;
  lu1=luc[1]-1; lu2=luc[2]-1;
  printf("%5d/%5d/%12d#\n",lu1+1,lu2+1,luc[0]);
  for(l=0;l<4;l++){l4=l*4;
    printf("  ");
    for(n=0;n<4;n++){printf("%d",tlu[lu1][l4+n]);}
    printf("  ");
    for(n=0;n<4;n++){printf("%d",tlu[lu2][l4+n]);}
    printf(" ");
    for(n=1;n<5;n++){printf("%3d",anm[0][l4+n]);}
    printf("\n");
  }
}
//
```

The following two lists show the practical results of its execution.

```
** Make the Euler Units by /D4i **
    1/    3/   11/   17/   23/   31/   38/   44/   51/   57/   65/   80/
  0330  0321  0312  0303  0231  0213  0132  0123  0033  1320  1302  1230
  2112  3012  3021  3120  3102  3120  3201  3210  3210  3012  3030  3012
  1221  1230  2130  2031  1320  3120  2310  3210  2121  0321  2121  0321
  3003  2103  1203  1212  2013  0213  1023  0123  1302  2013  0213  2103
   88/   90/   96/  102/  108/  123/  129/  135/  150/  156/  162/  168/
  1221  1212  1203  1122  1032  1023  2310  2301  2211  2130  2121  2112
  3003  3030  3021  3210  3120  3201  3021  3030  3120  3021  3030  3003
  0330  2121  2130  2301  2211  2310  0312  1212  1302  0312  1212  0330
  2112  0303  0312  0033  0303  0132  1023  0123  0033  1203  0303  1221
  170/  178/  193/  201/  207/  214/  220/  227/  235/  241/  247/  255/
  2103  2031  2013  3300  3210  3201  3120  3102  3030  3021  3012  3003
  3012  3120  3102  2031  2031  3021  2031  3012  2121  3021  3012  2112
  1230  1302  1320  0213  1302  0312  1302  0321  0303  0312  0321  1221
  0321  0213  0231  1122  0123  0132  0213  0231  1212  0312  0321  0330
 [Total Count of Euler Units = 256]

** List of Reconstruction: Standard Magic Squares of Order 4 **
    (Standard Solutions with Used_Units: H/L/ and Sol_Number#)
```

```
  1/   20/           1#      2/   20/          21#      3/   10/          41#
0330 0303  1 16 13  4    0330 0303  1 16 13  4    0321 0321  1 16 11  6
2112 2121 11  6  7 10    1221 2121  7 10 11  6    3012 0321 13  4  7 10
1221 3030  8  9 12  5    2112 3030 12  5  8  9    1230 3012  8  9 14  3
3003 1212 14  3  2 15    3003 1212 14  3  2 15    2103 3012 12  5  2 15

  4/   30/           73#      5/   20/          85#      6/    7/          93#
0321 0231  1 15 12  6    0321 0303  1 16  9  8    0321 0321  1 16 11  6
3102 0321 13  8  3 10    3210 2121 15 10  7  2    2103 1230 10  7  4 13
0231 3012  4  9 14  7    0123 3030  4  5 12 13    1230 3012  8  9 14  3
3012 3102 16  2  5 11    3012 1212 14  3  6 11    3012 2103 15  2  5 12

  7/    6/          117#      8/   26/         141#      9/   17/         153#
0321 0321  1 16 11  6    0321 0231  1 15 12  6    0321 0303  1 16  9  8
1230 2103  7 10 13  4    1320 2103  7 14  9  4    0213 3120  4 10  7 13
3012 1230 14  3  8  9    2013 1230 10  3  8 13    3120 2031 15  5 12  2
2103 3012 12  5  2 15    3012 3102 16  2  5 11    3012 1212 14  3  6 11

 10/    3/          161#      12/   30/         201#      14/   26/         213#
0321 0321  1 16 11  6    0312 0231  1 15  8 10    0312 0231  1 15  8 10
0321 3012  4 13 10  7    3201 0321 13 12  3  6    2310 2103 11 14  5  4
3012 1230 14  3  8  9    0132 3012  4  5 14 11    1023 1230  6  3 12 13
3012 2103 15  2  5 12    3021 3102 16  2  9  7    3021 3102 16  2  9  7

 15/    7/          225#      16/    3/         249#      23/    7/         289#
0312 0321  1 16  7 10    0312 0321  1 16  7 10    0231 0321  1 12 15  6
1203 1230  6 11  4 13    0312 3012  4 13  6 11    3102 1230 14  7  4  9
2130 3012 12  5 14  3    3021 1230 14  3 12  5    1320 3012  8 13 10  3
3021 2103 15  2  9  8    3021 2103 15  2  9  8    2013 2103 11  2  5 16

 24/   10/          313#      25/   55/         353#      26/    8/         361#
0231 0321  1 12 15  6    0231 0033  1  9 16  8    0231 0321  1 12 15  6
3102 0321 13  8  3 10    3210 2301 15 12  5  2    2103 1320 10  8  3 13
0231 3012  4  9 14  7    0123 3210  4  7 10 13    1230 2013  7  9 14  4
3102 3012 16  5  2 11    3102 1122 14  6  3 11    3102 3012 16  5  2 11

 27/    3/          373#      28/    6/         405#      29/   52/         429#
0231 0321  1 12 15  6    0231 0321  1 12 15  6    0231 0033  1  9 16  8
1320 3012  8 13 10  3    1320 2103  7 14  9  4    0213 3300  4 12  5 13
3102 1230 14  7  4  9    2013 1230 10  3  8 13    3120 2211 15  7 10  2
2013 2103 11  2  5 16    3102 3012 16  5  2 11    3102 1122 14  6  3 11

 30/    4/          437#      35/    1/         449#      37/    1/         453#
0231 0321  1 12 15  6    0213 0330  1 12  8 13    0213 0330  1 12  8 13
0321 3102  4 14  9  7    2301 2112 11 14  2  7    1302 2112  7 14  2 11
3012 0231 13  3  8 10    1032 1221  6  3 15 10    2031 1221 10  3 15  6
3102 3012 16  5  2 11    3120 3003 16  5  9  4    3120 3003 16  5  9  4

 39/   10/          461#      41/    6/         501#      42/    8/         525#
0132 0321  1  8 15 10    0132 0321  1  8 15 10    0132 0321  1  8 15 10
3201 0321 13 12  3  6    2310 2103 11 14  5  4    1203 1320  6 12  3 13
0132 3012  4  5 14 11    1023 1230  6  3 12 13    2130 2013 11  5 14  4
3201 3012 16  9  2  7    3201 3012 16  9  2  7    3201 3012 16  9  2  7

 43/    4/          537#      48/    1/         549#      50/    1/         557#
0132 0321  1  8 15 10    0123 0330  1  8 12 13    0123 0330  1  8 12 13
0312 3102  4 14  5 11    2301 2112 11 14  2  7    1302 2112  7 14  2 11
3021 0231 13  3 12  6    1032 1221  6  3 15 10    2031 1221 10  3 15  6
3201 3012 16  9  2  7    3210 3003 16  9  5  4    3210 3003 16  9  5  4

 68/   10/          561#      69/    1/         565#      76/    8/         569#
1302 0321  5 16  3 10    1302 0330  5 16  4  9    1302 0321  5 16  3 10
3120 0321 13  8 11  2    3210 2112 15 10  6  3    0123 1320  2  8 11 13
0033 3012  4  1 14 15    0123 1221  2  7 11 14    3030 2013 15  1 14  4
2211 3012 12  9  6  7    2031 3003 12  1 13  8    2211 3012 12  9  6  7

 77/    1/          573#      88/   20/         581#      89/   20/         601#
1302 0330  5 16  4  9    1221 0303  5 12  9  8    1221 0303  5 12  9  8
0213 2112  3 10  6 15    3003 2121 15  2  3 14    0330 2121  3 14 15  2
3120 1221 14  7 11  2    0330 3030  4 13 16  1    3003 3030 16  1  4 13
2031 3003 12  1 13  8    2112 1212 10  7  6 11    2112 1212 10  7  6 11
```

```
 91/   26/          621#     94/   24/          625#    103/   10/          629#
1212  0231   5 11  8 10    1212  0231   5 11  8 10    1122  0321   5  8 11 10
3120  2103  15  6  9  4    0123  3102   4  6  9 15    3300  0321  13 16  3  2
0033  1230   2  3 16 13    3030  0231  13  3 16  2    0213  3012   4  9  6 15
2301  3102  12 14  1  7    2301  3102  12 14  1  7    2031  3012  12  1 14  7

106/    8/          633#    111/    1/          637#    112/   26/          645#
1122  0321   5  8 11 10    1032  0330   5  4 16  9    1032  0231   5  3 16 10
0303  1320   2 16  3 13    3210  2112  15 10  6  3    3300  2103  15 14  1  4
3210  2013  15  9  6  4    0123  1221   2  7 11 14    0213  1230   2 11  8 13
2031  3012  12  1 14  7    2301  3003  12 13  1  8    2121  3102  12  6  9  7
[Solution Counts = 656]
```

Why! We could have reconstructed only these 656 solutions among the 880 standard solutions in total. What did we fail to notice?

9. Now we must step back again to the beginning point: the basic fact.

I mean we must observe the decomposition diagrams of the original MS44 very carefully before all and find anything irregular in their contents.

```
** Standard Magic Squares of Order 4 Composed by the Ordinary
    Method: with Decomposed Units by N4i and Check Sums of Each **
          1#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1 16 13  4|34|34|   0 15 12  3|30|30|  0 3 3 0|6|6|  0 3 0 3|6|6|
12  9  8  5|34|34|  11  8  7  4|30|30|  2 2 1 1|6|6|  3 0 3 0|6|6|
 6  7 10 11|34|34|   5  6  9 10|30|30|  1 1 2 2|6|6|  1 2 1 2|6|6|
15  2  3 14|34|34|  14  1  2 13|30|30|  3 0 0 3|6|6|  2 1 2 1|6|6|

         24#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1 16  5 12|34|34|   0 15  4 11|30|30|  0 3 1 2|6|6|  0 3 0 3|6|6|
10 13  4  7|34|34|   9 12  3  6|30|30|  2 3 0 1|6|6|  1 0 3 2|6|6|
 8  3 14  9|34|34|   7  2 13  8|30|30|  1 0 3 2|6|6|  3 2 1 0|6|6|
15  2 11  6|34|34|  14  1 10  5|30|30|  3 0 2 1|6|6|  2 1 2 1|6|6|

         26#\34/34|  /Math      \30/30| /D4i   H\6/6|       L\6/6|
 1 15 14  4|34|34|   0 14 13  3|30|30|  0 3 3 0|6|6|  0 2 1 3|6|6|
12  6  7  9|34|34|  11  5  6  8|30|30|  2 1 1 2|6|6|  3 1 2 0|6|6|
 8 10 11  5|34|34|   7  9 10  4|30|30|  1 2 2 1|6|6|  3 1 2 0|6|6|
13  3  2 16|34|34|  12  2  1 15|30|30|  3 0 0 3|6|6|  0 2 1 3|6|6|

         46#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1 15 10  8|34|34|   0 14  9  7|30|30|  0 3 2 1|6|6|  0 2 1 3|6|6|
16  6  3  9|34|34|  15  5  2  8|30|30|  3 1 0 2|6|6|  3 1 2 0|6|6|
 5 11 14  4|34|34|   4 10 13  3|30|30|  1 2 3 0|6|6|  0 2 1 3|6|6|
12  2  7 13|34|34|  11  1  6 12|30|30|  2 0 1 3|6|6|  3 1 2 0|6|6|

         62#\34/34|  /Math      \30/30| /D4i   H\6/6|       L\6/6|
 1 14 15  4|34|34|   0 13 14  3|30|30|  0 3 3 0|6|6|  0 1 2 3|6|6|
12  7  6  9|34|34|  11  6  5  8|30|30|  2 1 1 2|6|6|  3 2 1 0|6|6|
 8 11 10  5|34|34|   7 10  9  4|30|30|  1 2 2 1|6|6|  3 2 1 0|6|6|
13  2  3 16|34|34|  12  1  2 15|30|30|  3 0 0 3|6|6|  0 1 2 3|6|6|

         78#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1 14 11  8|34|34|   0 13 10  7|30|30|  0 3 2 1|6|6|  0 1 2 3|6|6|
16  5  4  9|34|34|  15  4  3  8|30|30|  3 1 0 2|6|6|  3 0 3 0|6|6|
 7 12 13  2|34|34|   6 11 12  1|30|30|  1 2 3 0|6|6|  2 3 0 1|6|6|
10  3  6 15|34|34|   9  2  5 14|30|30|  2 0 1 3|6|6|  1 2 1 2|6|6|

         94#\34/34|  /Math      \30/30| /D4i   H\6/6|       L\6/6|
 1 13 16  4|34|34|   0 12 15  3|30|30|  0 3 3 0|6|6|  0 0 3 3|6|6|
12  8  5  9|34|34|  11  7  4  8|30|30|  2 1 1 2|6|6|  3 3 0 0|6|6|
 7 11 10  6|34|34|   6 10  9  5|30|30|  1 2 2 1|6|6|  2 2 1 1|6|6|
14  2  3 15|34|34|  13  1  2 14|30|30|  3 0 0 3|6|6|  1 1 2 2|6|6|
```

```
       115#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1 12 15  6|34|34|   0 11 14  5|30|30|  0 2 3 1|6|6|  0 3 2 1|6|6|
14  7  4  9|34|34|  13  6  3  8|30|30|  3 1 0 2|6|6|  1 2 3 0|6|6|
 8 13 10  3|34|34|   7 12  9  2|30|30|  1 3 2 0|6|6|  3 0 1 2|6|6|
11  2  5 16|34|34|  10  1  4 15|30|30|  2 0 1 3|6|6|  2 1 0 3|6|6|
       147#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1 11 16  6|34|34|   0 10 15  5|30|30|  0 2 3 1|6|6|  0 2 3 1|6|6|
14  8  3  9|34|34|  13  7  2  8|30|30|  3 1 0 2|6|6|  1 3 2 0|6|6|
 7 13 10  4|34|34|   6 12  9  3|30|30|  1 3 2 0|6|6|  2 0 1 3|6|6|
12  2  5 15|34|34|  11  1  4 14|30|30|  2 0 1 3|6|6|  3 1 0 2|6|6|
       151#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1 11 14  8|34|34|   0 10 13  7|30|30|  0 2 3 1|6|6|  0 2 1 3|6|6|
16  5  4  9|34|34|  15  4  3  8|30|30|  3 1 0 2|6|6|  3 0 3 0|6|6|
 7 12 13  2|34|34|   6 11 12  1|30|30|  1 2 3 0|6|6|  2 3 0 1|6|6|
10  6  3 15|34|34|   9  5  2 14|30|30|  2 1 0 3|6|6|  1 1 2 2|6|6|
       158#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1 10 16  7|34|34|   0  9 15  6|30|30|  0 2 3 1|6|6|  0 1 3 2|6|6|
15  8  2  9|34|34|  14  7  1  8|30|30|  3 1 0 2|6|6|  2 3 1 0|6|6|
 6 13 11  4|34|34|   5 12 10  3|30|30|  1 3 2 0|6|6|  1 0 2 3|6|6|
12  3  5 14|34|34|  11  2  4 13|30|30|  2 0 1 3|6|6|  3 2 0 1|6|6|
       172#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1  9 16  8|34|34|   0  8 15  7|30|30|  0 2 3 1|6|6|  0 0 3 3|6|6|
15 12  5  2|34|34|  14 11  4  1|30|30|  3 2 1 0|6|6|  2 3 0 1|6|6|
 4  7 10 13|34|34|   3  6  9 12|30|30|  0 1 2 3|6|6|  3 2 1 0|6|6|
14  6  3 11|34|34|  13  5  2 10|30|30|  3 1 0 2|6|6|  1 1 2 2|6|6|
       177#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1  8 15 10|34|34|   0  7 14  9|30|30|  0 1 3 2|6|6|  0 3 2 1|6|6|
13 12  3  6|34|34|  12 11  2  5|30|30|  3 2 0 1|6|6|  0 3 2 1|6|6|
 4  5 14 11|34|34|   3  4 13 10|30|30|  0 1 3 2|6|6|  3 0 1 2|6|6|
16  9  2  7|34|34|  15  8  1  6|30|30|  3 2 0 1|6|6|  3 0 1 2|6|6|
       194#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1  7 16 10|34|34|   0  6 15  9|30|30|  0 1 3 2|6|6|  0 2 3 1|6|6|
14 12  3  5|34|34|  13 11  2  4|30|30|  3 2 0 1|6|6|  1 3 2 0|6|6|
 4  6 13 11|34|34|   3  5 12 10|30|30|  0 1 3 2|6|6|  3 1 0 2|6|6|
15  9  2  8|34|34|  14  8  1  7|30|30|  3 2 0 1|6|6|  2 0 1 3|6|6|
       203#\34/34|  /Math      \30/30| /D4i   H\6/6|        L\6/6|
 1  6 16 11|34|34|   0  5 15 10|30|30|  0 1 3 2|6|6|  0 1 3 2|6|6|
15 12  2  5|34|34|  14 11  1  4|30|30|  3 2 0 1|6|6|  2 3 1 0|6|6|
 4  7 13 10|34|34|   3  6 12  9|30|30|  0 1 3 2|6|6|  3 2 0 1|6|6|
14  9  3  8|34|34|  13  8  2  7|30|30|  3 2 0 1|6|6|  1 0 2 3|6|6|
       207#\34/34|  /Math      \30/30| /D4i   H\7/5|       L\2/10|
 1  5 16 12|34|34|   0  4 15 11|30|30|  0 1 3 2|6|6|  0 0 3 3|6|6|
10 14  3  7|34|34|   9 13  2  6|30|30|  2 3 0 1|6|6|  1 1 2 2|6|6|
 8  4 13  9|34|34|   7  3 12  8|30|30|  1 0 3 2|6|6|  3 3 0 0|6|6|
15 11  2  6|34|34|  14 10  1  5|30|30|  3 2 0 1|6|6|  2 2 1 1|6|6|
. . . . .
```

(1) Any number of (0, 1, 2 and 3) comes just 4 times in each unit as often as the others. I found nothing irregular there.

(2) Every 4 numbers in any row and in any column always add up to the constant sum 6. Nothing irregular.

(3) But in the 2 primary diagonals in many decomposed units I found something irregular. Every 4 numbers on them do not always add up to the constant sum 6. While they usually add up to 6 in many cases, they sometimes add up to 10, 7, 5 or 2, to my surprise. What do they mean?

I found the sums of primary diagonals in higher units are only 5, 6 or 7. The sums in lower units are only 2, 6 or 10, and nothing else. And it seems there are some clear correspondences between higher units and lower.

I found there are only three cases possible: (5, 10), (6, 6) and (7, 2);

[1] 5 x 4 + 10 = 30; (/High) (/Low) [2] 6 x 4 + 6 = 30; (/High) (/Low) [3] 7 x 4 + 2 = 30; (/High) (/Low)

'30' is the constant value of every sum of 4 numbers on each row, column and diagonal in the mathematical notation of objective magic squares 4x4.

They mean it is all right if the sums of essential parts are equal to the constant value 30 on the whole, and they don't care whether the component sums in each Euler Units really add up to 6 or not.

How can we define the new rules above with these exceptional irregularities for our Euler Units to produce them?

10. The following lists are my answers to that question. They contain the program I wrote and the results of my recent efforts.

I could not help dividing the making process of Euler units into two steps for higher and lower, and defined them separately at last.

```
//** 'New-Euler Type' of Standard Magic Squares 4x4 **
//**  Composed by "New Euler's Method" with /D4i **
//** 'NewEulerMS44D4i.c' Dictated by Kanji Setsuda **
//** on Nov.24, 2011 and Revised on Jan.23, 2012 **
//** Working with MacOSX 10.7.2 & Xcode 4.2.1 **
//
#include <stdio.h>
//
short int cnt, hcnt, lcnt;
short cnt2, cnt3;
short luc[13];
short u1,u2;
short LSM;
short nm[17], uflg[17];
short olst[881][17];
short onm[3][50];
short anm[13][30];
short thlu[189][17], tllu[769][17];
short mtc[189][769];
//
void hstp01(void), hstp02(void), hstp03(void), hstp04(void), hstp05(void);
void hstp06(void), hstp07(void), hstp08(void), hstp09(void), hstp10(void);
void hstp11(void), hstp12(void), hstp13(void), hstp14(void), hstp15(void);
void hstp16(void);
void hansrecord(void);
//
void prlunt(short x);
//
void lstp01(void), lstp02(void), lstp03(void), lstp04(void), lstp05(void);
void lstp06(void), lstp07(void), lstp08(void), lstp09(void), lstp10(void);
void lstp11(void), lstp12(void), lstp13(void), lstp14(void), lstp15(void);
void lstp16(void);
void lansrecord(void);
//
void tblmtc(void);
void cmbcmp(void);
void recprans(void), pr3ans(void), pr1ans(void);
//
//* Main Program *
int main(void){
```

```
 short n;
  printf("\n");
  printf("*** The 'Standard' Magic Squares 4x4 made by /D4i ***\n");
  for(n=0;n<17;n++){nm[n]=0;}
  for(n=0;n<4;n++){uflg[n]=0;}
  LSM=6; cnt=0; cnt3=0;
  printf(" ** Make Euler Units for High Positions by /D4i **\n");
  hstp01();                       // Make the Euler Units for High
  if(cnt3>0){prlunt(cnt3);}
  hcnt=cnt; printf(" [Total Count of Euler Units for High = %d]\n",hcnt);
  for(n=0;n<17;n++){nm[n]=0;}
  for(n=0;n<4;n++){uflg[n]=0;}
  cnt=0; cnt3=0;
  printf("\n");
  printf(" ** Make Euler Units for Low Positions by /D4i **\n");
  lstp01();                       // Make the Euler Units for Low
  if(cnt3>0){prlunt(cnt3);}
  lcnt=cnt; printf(" [Total Count of Euler Units for Low = %d]\n",lcnt);
  tblmtc();
  printf("\n");
  printf("** List of Standard Type of Magic Squares of Order 4 **\n");
  printf("** Standard Solutions with D4i: H/L/ and Sol_Number# **\n");
  lcnt=cnt; cnt=0; cnt3=0;
  cmbcmp();                       // Combine, Compose and Print
  if(cnt3==1){pr1ans();}          // Print the Rest One
  printf(" [Solution Counts = %d]\n",cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on Jan.22, 2012 with MacOSX 10.7.2 & Xcode 4.2.1 **\n");
  printf("\n");
 return 0;
}
//
// Make the Euler Units for High Positions
// Level #1:
// Set n1
void hstp01(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[1]=i;
      uflg[i]++; hstp02(); uflg[i]--;}
  }
}
// Set n2
void hstp02(){
 short i;
  for(i=3;i>=0;i--){
    if(uflg[i]<4){nm[2]=i;
      uflg[i]++; hstp03(); uflg[i]--;}
  }
}
// Set n3
void hstp03(){
 short i;
  for(i=0;i<4;i++){cnt2=0;
    if(uflg[i]<4){nm[3]=i;
      uflg[i]++; hstp04(); uflg[i]--;}
  }
}
// Set n4=LSM-n1-n2-n3 & n4>=n1
void hstp04(){
 short i;
  i=LSM-nm[1]-nm[2]-nm[3];
  if((nm[1]<=i)&&(i<=3)){
    if(uflg[i]<4){nm[4]=i;
```

```
      uflg[i]++; hstp05(); uflg[i]--;}}
}
// Set n5
void hstp05(){
 short i;
  for(i=3;i>=0;i--){
    if(uflg[i]<4){nm[5]=i;
      uflg[i]++; hstp06(); uflg[i]--;}
  }
}
// Set n9
void hstp06(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[9]=i;
      uflg[i]++; hstp07(); uflg[i]--;}
  }
}
// Set n13=LSM-n1-n5-n9 & n13>=n4
void hstp07(){
 short i;
  i=LSM-nm[1]-nm[5]-nm[9];
  if((nm[4]<=i)&&(i<=3)){
    if(uflg[i]<4){nm[13]=i;
      uflg[i]++; hstp08(); uflg[i]--;}}
}
// Level #2:
// Set n6
void hstp08(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[6]=i;
      uflg[i]++; hstp09(); uflg[i]--;}
  }
}
// Set n7
void hstp09(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[7]=i;
      uflg[i]++; hstp10(); uflg[i]--;}
  }
}
// Set n8=LSM-n5-n6-n7
void hstp10(){
 short i;
  i=LSM-nm[5]-nm[6]-nm[7];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[8]=i;
      uflg[i]++; hstp11(); uflg[i]--;}}
}
// Set n10
void hstp11(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[10]=i;
      sm=nm[4]+nm[7]+nm[10]+nm[13];          //Check the sum of Primary Diagonal
      if((4<sm)&&(sm<8)){                    //All right if it is 5,6 or 7
        uflg[i]++; hstp12(); uflg[i]--;}}
  }
}
// Set n14=LSM-n2-n6-n10
void hstp12(){
 short i;
  i=LSM-nm[2]-nm[6]-nm[10];
```

```
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[14]=i;
      uflg[i]++; hstp13(); uflg[i]--;}}
}
// Level #3:
// Set n11
void hstp13(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[11]=i;
      uflg[i]++; hstp14(); uflg[i]--;}
  }
}
// Set n12=LSM-n9-n10-n11
void hstp14(){
 short i;
  i=LSM-nm[9]-nm[10]-nm[11];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[12]=i;
      uflg[i]++; hstp15(); uflg[i]--;}}
}
// Set n15=LSM-n3-n7-n11
void hstp15(){
 short i;
  i=LSM-nm[3]-nm[7]-nm[11];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[15]=i;
      uflg[i]++; hstp16(); uflg[i]--;}}
}
// Set n16=LSM-n4-n8-n12 & n16=LSM-n13-n14-n15 & n16>=n1
void hstp16(){
 short i,j,sm;
  i=LSM-nm[4]-nm[8]-nm[12];
  j=LSM-nm[13]-nm[14]-nm[15];
  if(i==j){
    if((nm[1]<=i)&&(i<=3)){
      if(uflg[i]<4){nm[16]=i;
        sm=nm[1]+nm[6]+nm[11]+nm[16];       //Check the sum of another Primary Diagonal
        if((4<sm)&&(sm<8)){                 //All right if it is 5,6 or 7
          uflg[i]++; hansrecord(); uflg[i]--;}}}}
}
//
// Record the Answers
void hansrecord(){
 short n;
  for(n=0;n<16;n++){thlu[cnt][n]=nm[n+1];}
  cnt++; cnt2++;
  if(cnt2==1){luc[cnt3]=cnt; luc[cnt3+1]=0;
    for(n=0;n<16;n++){anm[cnt3][n]=nm[n+1]; anm[cnt3+1][n]=0;}
    cnt3++; if(cnt3==11){prlunt(cnt3); cnt3=0;}
  }
}
//
// Print the List of Euler Units
void prlunt(short x){
 short l,l4,m,n;
  for(m=0;m<x;m++){printf("%5d/",luc[m]);}
  printf("\n");
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<x;m++){printf("  ");
      for(n=0;n<4;n++){printf("%d",anm[m][l4+n]);}}
    printf("\n");
  }
}
//
```

```
// Make the Euler Units for Low Positions
// Level #1:
// Set n1:
void lstp01(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[1]=i;
      uflg[i]++; lstp02(); uflg[i]--;}
  }
}
// Set n2
void lstp02(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[2]=i;
      uflg[i]++; lstp03(); uflg[i]--;}
  }
}
// Set n3
void lstp03(){
 short i;
  for(i=0;i<4;i++){cnt2=0;
    if(uflg[i]<4){nm[3]=i;
      uflg[i]++; lstp04(); uflg[i]--;}
  }
}
// Set n4=LSM-n1-n2-n3
void lstp04(){
 short i;
  i=LSM-nm[1]-nm[2]-nm[3];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[4]=i;
      uflg[i]++; lstp05(); uflg[i]--;}}
}
// Set n5
void lstp05(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[5]=i;
      uflg[i]++; lstp06(); uflg[i]--;}
  }
}
// Set n9
void lstp06(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[9]=i;
      uflg[i]++; lstp07(); uflg[i]--;}
  }
}
// Set n13=LSM-n1-n5-n9
void lstp07(){
 short i;
  i=LSM-nm[1]-nm[5]-nm[9];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[13]=i;
      uflg[i]++; lstp08(); uflg[i]--;}}
}
// Level #2:
// Set n6
void lstp08(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[6]=i;
      uflg[i]++; lstp09(); uflg[i]--;}
```

```
  }
}
// Set n7
void lstp09(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[7]=i;
      uflg[i]++; lstp10(); uflg[i]--;}
  }
}
// Set n8=LSM-n5-n6-n7
void lstp10(){
 short i;
  i=LSM-nm[5]-nm[6]-nm[7];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[8]=i;
      uflg[i]++; lstp11(); uflg[i]--;}}
}
// Set n10
void lstp11(){
 short i,sm;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[10]=i;
      sm=nm[4]+nm[7]+nm[10]+nm[13];       //Check the sum of Primary Diagonal
      if((sm%4)==2){                     //All right if it is 2,6 or 10
        uflg[i]++; lstp12(); uflg[i]--;}}
  }
}
// Set n14=LSM-n2-n6-n10
void lstp12(){
 short i;
  i=LSM-nm[2]-nm[6]-nm[10];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[14]=i;
      uflg[i]++; lstp13(); uflg[i]--;}}
}
// Level #3:
// Set n11
void lstp13(){
 short i;
  for(i=0;i<4;i++){
    if(uflg[i]<4){nm[11]=i;
      uflg[i]++; lstp14(); uflg[i]--;}
  }
}
// Set n12=LSM-n9-n10-n11
void lstp14(){
 short i;
  i=LSM-nm[9]-nm[10]-nm[11];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[12]=i;
      uflg[i]++; lstp15(); uflg[i]--;}}
}
// Set n15=LSM-n3-n7-n11
void lstp15(){
 short i;
  i=LSM-nm[3]-nm[7]-nm[11];
  if((0<=i)&&(i<=3)){
    if(uflg[i]<4){nm[15]=i;
      uflg[i]++; lstp16(); uflg[i]--;}}
}
// Set n16=LSM-n4-n8-n12
void lstp16(){
 short i,j,sm;
  i=LSM-nm[4]-nm[8]-nm[12];
```

```
  j=LSM-nm[13]-nm[14]-nm[15];
  if(i==j){
    if((0<=i)&&(i<=3)){
      if(uflg[i]<4){nm[16]=i;
        sm=nm[1]+nm[6]+nm[11]+nm[16];     //Check the sum of another Primary Diagonal
        if((sm%4)==2){                    //All right if it is 2,6 or 10
          uflg[i]++; lansrecord(); uflg[i]--;}}}}
}
//
// Record the Answers
void lansrecord(){
 short n;
  for(n=0;n<16;n++){tllu[cnt][n]=nm[n+1];}
  cnt++; cnt2++;
  if(cnt2==1){luc[cnt3]=cnt; luc[cnt3+1]=0;
    for(n=0;n<16;n++){anm[cnt3][n]=nm[n+1]; anm[cnt3+1][n]=0;}
    cnt3++; if(cnt3==11){prlunt(cnt3); cnt3=0;}
  }
}
//
// Check how well they match and Save the Data to the Table
void tblmtc(){
 short h,l,n,d;
  for(h=0;h<hcnt;h++){
    for(l=0;l<lcnt;l++){d=0;
      for(n=0;n<16;n++){if(thlu[h][n]==tllu[l][n]){d++;}}
      mtc[h][l]=d;}
  }
}
//
//* Combine, Compose and Print *
void cmbcmp(){
 short n,d,fc;
 short s1,s2,s3,s4;
  for(u1=0;u1<hcnt;u1++){cnt2=0;
    s1=thlu[u1][0]+thlu[u1][5]+thlu[u1][10]+thlu[u1][15];
    s2=thlu[u1][3]+thlu[u1][6]+thlu[u1][9]+thlu[u1][12];
    for(u2=0;u2<lcnt;u2++){
      s3=tllu[u2][0]+tllu[u2][5]+tllu[u2][10]+tllu[u2][15];
      s4=tllu[u2][3]+tllu[u2][6]+tllu[u2][9]+tllu[u2][12];
      if((s1*4+s3==30)&&(s2*4+s4==30)){   //Check the sums of Primary Diagnals in total
        if(mtc[u1][u2]==4){
          for(n=1;n<17;n++){uflg[n]=0;}
          for(n=0;n<16;n++){
            d=thlu[u1][n]*4+tllu[u2][n]+1;
            nm[n+1]=d; uflg[d]++;
          }
          fc=0;
          for(n=1;n<17;n++){if(uflg[n]==0){fc++;}}
          if(fc==0){
            if((nm[1]<nm[4])&&(nm[4]<nm[13])&&(nm[1]<nm[16])){
              recprans();}}
        }}
    }
  }
}
//
// Record and Print the Solution List
void recprans(){
 short n;
  cnt++; cnt2++;
  if(cnt2==1){
    luc[cnt3*3]=cnt;
    for(n=1;n<17;n++){anm[cnt3][n]=nm[n]; anm[cnt3+1][n]=0;}
    luc[cnt3*3+1]=u1+1; luc[cnt3*3+2]=u2+1;
```

```
    cnt3++; if(cnt3==3){pr3ans(); cnt3=0;}
  }
}
//
// Print 3 Answers
void pr3ans(){
 short lu1,lu2;
 short l,l4,m,n;
  printf("%5d/%5d/%12d#%7d/%5d/%12d#%7d/%5d/%12d#\n",
         luc[1],luc[2],luc[0],luc[4],luc[5],luc[3],luc[7],luc[8],luc[6]);
  for(l=0;l<4;l++){l4=l*4;
    for(m=0;m<3;m++){
      lu1=luc[m*3+1]-1; lu2=luc[m*3+2]-1;
      printf("  ");
      for(n=0;n<4;n++){printf("%d",thlu[lu1][l4+n]);}
      printf("  ");
      for(n=0;n<4;n++){printf("%d",tllu[lu2][l4+n]);}
      printf(" ");
      for(n=1;n<5;n++){printf("%3d",anm[m][l4+n]);}
      if(m<2){printf("  ");}
    }
    printf("\n");
  }
}
//
// Print the Rest One
void pr1ans(){
 short lu1,lu2;
 short l,l4,n;
  lu1=luc[1]-1; lu2=luc[2]-1;
  printf("%5d/%5d/%12d#\n",lu1+1,lu2+1,luc[0]);
  for(l=0;l<4;l++){l4=l*4;
    printf("  ");
    for(n=0;n<4;n++){printf("%d",thlu[lu1][l4+n]);}
    printf("  ");
    for(n=0;n<4;n++){printf("%d",tllu[lu2][l4+n]);}
    printf(" ");
    for(n=1;n<5;n++){printf("%3d",anm[0][l4+n]);}
    printf("\n");
  }
}
//
```

*** The 'Standard' Magic Squares 4x4 made by New Euler's Method ***

** Make **Euler Units** for **High** Positions by /D4i **

1/	3/	21/	48/	58/	67/	69/	96/	105/	123/	125/
0303	0312	0321	0330	0213	0222	0231	0123	0132	0033	1302
2211	3102	3012	3012	3201	3300	3102	3201	3102	2301	3102
1032	0231	0231	1221	0132	1113	0132	0132	0231	1122	0231
3120	3021	3102	2103	3120	2031	3201	3210	3201	3210	2031

144/	148/	154/	160/	166/	170/
1311	1212	1221	1122	1131	1032
3030	3003	3003	3003	3210	3102
0123	0330	0330	0330	0303	0231
2202	2121	2112	2211	2022	2301

[Total Count of Euler Units for High = 188]

** Make **Euler Units** for **Low** Positions by /D4i **

1/	17/	44/	63/	90/	92/	117/	133/	152/	177/	193/
0033	0123	0132	0213	0222	0231	0303	0312	0321	0330	1023
1122	0123	0132	0123	2130	0213	1032	0132	0123	1023	0312

```
2211  3210  3201  3210  3003  3102  2121  3201  3012  2112  2310
3300  3210  3201  3120  1311  3120  3210  3021  3210  3201  3021

212/  239/  241/  257/  259/  278/  294/  306/  331/  358/  360/
1032  1113  1122  1131  1203  1212  1221  1230  1302  1311  1320
0123  0330  0033  2310  0132  0123  0132  0123  0033  0132  0123
2301  3201  2211  3003  2130  2301  2310  2103  2211  3003  2103
3210  2022  3300  0222  3201  3030  3003  3210  3120  2220  3120

385/  410/  412/  439/  464/  476/  492/  511/  513/  529/  531/
2013  2022  2031  2103  2112  2121  2130  2202  2211  2220  2301
0213  0330  0213  0123  0330  0033  0312  0330  0033  0312  0123
1320  1203  1122  1230  1203  1302  1203  3021  1302  3003  1032
3120  3111  3300  3210  3021  3210  3021  1113  3120  1131  3210

558/  577/  593/  618/  637/  653/  678/  680/  707/  726/  753/
2310  3003  3012  3021  3030  3102  3111  3120  3201  3210  3300
0132  0132  0123  0132  0123  0123  0330  0213  0132  0123  0033
1023  1221  0321  0312  1212  0321  1023  0213  0312  0123  1122
3201  2310  3210  3201  2301  3120  2202  3120  3021  3210  2211
[Total Count of Euler Units for Low = 768]
```

So many! We have to make 188 x 768 combinations of units to examine and select the 880 correct solutions out of 144384 cases. What a terrible job!

```
** List of Standard Type of Magic Squares of Order 4 **
** Standard Solutions with D4i: H/L/ and Sol_Number# **
  1/  561/          1#    2/  571/          2#    3/  518/          3#
0303  2310   3 16  2 13  0303  2310   3 16  2 13  0312  2211   3 15  6 10
2211  0312   9 12  6  7  1212  2310   7 12  6  9  3102  0303  13  8  1 12
1032  3021   8  1 15 10  2031  1023  10  1 15  8  0231  3030   4  9 16  5
3120  1023  14  5 11  4  3120  1023  14  5 11  4  3021  1122  14  2 11  7

  4/  15/           7#    5/  491/          19#    7/  526/          23#
0312  0033   1 13  8 12  0312  2121   3 14  7 10  0312  2211   3 15  6 10
3201  3210  16 11  2  5  3210  3210  16 11  6  1  2103  3300  12  8  1 13
0132  2121   3  6 15 10  0123  1032   2  5 12 15  1230  0033   5  9 16  4
3021  1302  14  4  9  7  3021  0303  13  4  9  8  3021  1122  14  2 11  7

  8/  18/           27#   9/  47/           31#   12/  37/           43#
0312  0123   1 14  7 12  0312  0132   1 14  8 11  0312  0123   1 14  7 12
2301  0231   9 15  4  6  2310  1203  10 15  5  4  1203  3210   8 11  2 13
1032  3102   8  2 13 11  1023  2130   7  2 12 13  2130  1032  10  5 16  3
3021  3210  16  3 10  5  3021  3201  16  3  9  6  3021  2301  15  4  9  6

 13/  21/           67#   14/  518/          71#   17/  486/          75#
0312  0123   1 14  7 12  0312  2211   3 15  6 10  0312  2121   3 14  7 10
1302  1230   6 15  4  9  1320  0303   5 16  9  4  0213  1212   2 11  6 15
2031  2103  11  2 13  8  2013  3030  12  1  8 13  3120  3030  16  5 12  1
3021  3210  16  3 10  5  3021  1122  14  2 11  7  3021  0303  13  4  9  8

 18/  10/           79#   19/  526/          119#  21/  432/          125#
0312  0033   1 13  8 12  0312  2211   3 15  6 10  0321  2031   3 13 12  6
0312  2211   3 15  6 10  0321  3300   4 16  9  5  3012  2301  15  4  5 10
3021  3120  16  2 11  5  3012  0033  13  1  8 12  0231  1032   2  9 16  7
3021  1302  14  4  9  7  3021  1122  14  2 11  7  3102  1302  14  8  1 11

 22/  15/           128#  25/   6/           140#  26/  121/          144#
0321  0033   1 13 12  8  0321  0033   1 13 12  8  0321  0303   1 16  9  8
3102  3210  16  7  2  9  3201  2121  15 10  3  6  3210  1212  14 11  6  3
0231  2121   3 10 15  6  0132  1212   2  7 14 11  0123  3030   4  5 12 13
3012  1302  14  4  5 11  3012  3300  16  4  5  9  3012  2121  15  2  7 10

 28/  13/           152#  29/  250/          184#  30/  38/           187#
0321  0033   1 13 12  8  0321  1122   2 14 11  7  0321  0123   1 14 11  8
3012  3300  16  4  5  9  3012  2301  15  4  5 10  3102  3030  16  5  4  9
```

```
1230  1122   6 10 15  3    1320  3030   8 13 12  1    1230  2301   7 12 13  2
2103  2211  11  7  2 14    2013  0213   9  3  6 16    2013  1212  10  3  6 15

 31/  287/         191#     32/   37/         195#     34/    6/         219#
0321  1212   2 15 10  7    0321  0123   1 14 11  8    0321  0033   1 13 12  8
2013  2301  11  4  5 14    2103  3210  12  7  2 13    2310  2121  11 14  7  2
1230  3030   8  9 16  1    1230  1032   6  9 16  3    1023  1212   6  3 10 15
3102  0123  13  6  3 12    3012  2301  15  4  5 10    3012  3300  16  4  5  9

 36/    1/         223#     37/  550/         227#     38/   47/         231#
0321  0033   1 13 12  8    0321  2301   3 16  9  6    0321  0132   1 14 12  7
1203  1122   6 10  3 15    1230  2301   7 12 13  2    1320  1203   6 15  9  4
2130  2211  11  7 14  2    2013  1032  10  1  8 15    2013  2130  11  2  8 13
3012  3300  16  4  5  9    3102  1032  14  5  4 11    3012  3201  16  3  5 10

 39/   31/         243#     40/  256/         267#     41/   29/         271#
0321  0123   1 14 11  8    0321  1122   2 14 11  7    0321  0123   1 14 11  8
1230  2301   7 12 13  2    1230  3300   8 12 13  1    1320  2031   7 13 12  2
3012  3210  16  3  6  9    3102  2031  15  5  4 10    3012  3300  16  4  5  9
2103  1032  10  5  4 15    2013  0213   9  3  6 16    2013  1212  10  3  6 15

 43/  129/         274#     44/  545/         282#     46/    1/         285#
0321  0303   1 16  9  8    0321  2301   3 16  9  6    0321  0033   1 13 12  8
0213  3210   4 11  6 13    0231  1302   2 12 13  7    0312  1122   2 14  7 11
3120  1032  14  5 12  3    3012  2031  15  1  8 10    3021  2211  15  3 10  6
3012  2121  15  2  7 10    3102  1032  14  5  4 11    3012  3300  16  4  5  9

 47/   10/         291#     50/   13/         331#     52/   12/         351#
0321  0033   1 13 12  8    0330  0033   1 13 16  4    0330  0033   1 13 16  4
0321  2211   3 15 10  6    2112  3300  12  8  5  9    2211  3120  12 10  7  5
3012  3120  16  2  7  9    1221  1122   6 10 11  7    1122  1302   6  8  9 11
3012  1302  14  4  5 11    3003  2211  15  3  2 14    3003  2211  15  3  2 14

 54/    4/         359#     56/   13/         367#     58/  313/         387#
0330  0033   1 13 16  4    0330  0033   1 13 16  4    0213  1230   2 11  8 13
1212  1122   6 10  7 11    1221  3300   8 12  9  5    3201  1302  14 12  1  7
2121  3300  12  8  9  5    2112  1122  10  6  7 11    0132  2031   3  5 16 10
3003  2211  15  3  2 14    3003  2211  15  3  2 14    3120  2103  15  6  9  4

 59/  179/         388#     60/  561/         392#     61/  313/         393#
0213  0330   1 12  8 13    0213  2310   3 12  6 13    0213  1230   2 11  8 13
2301  1221  10 15  3  6    2301  0312   9 16  2  7    2310  1302  10 16  5  3
1032  2112   7  2 14 11    1122  3021   8  5 11 10    1023  2031   7  1 12 14
3120  3003  16  5  9  4    3030  1023  14  1 15  4    3120  2103  15  6  9  4

 62/  319/         394#     63/   98/         395#     64/  571/         403#
0213  1230   2 11  8 13    0213  0231   1 11  8 14    0213  2310   3 12  6 13
1203  2301   7 12  1 14    1302  1320   6 16  3  9    1302  2310   7 16  2  9
2130  1032  10  5 16  3    2031  3102  12  2 13  7    2121  1023  10  5 11  8
3120  2103  15  6  9  4    3120  2013  15  5 10  4    3030  1023  14  1 15  4
. . . . .
171/  100/         851#     172/   98/         852#     173/  145/         860#
1032  0231   5  3 16 10    1032  0231   5  3 16 10    1032  0312   5  4 14 11
3201  2031  15  9  4  6    3210  1320  14 12  7  1    3300  3021  16 13  3  2
0132  1302   2  8 13 11    0123  3102   4  6  9 15    0123  0312   1  8 10 15
2301  3102  12 14  1  7    2301  2013  11 13  2  8    2211  3021  12  9  7  6

174/   47/         861#     177/  508/         865#     178/  100/         867#
1032  0132   5  2 16 11    1032  2130   7  2 16  9    1032  0231   5  3 16 10
3300  1203  14 15  1  4    2301  3201  12 15  1  6    2310  2031  11 13  8  2
0213  2130   3 10  8 13    1302  0132   5 14  4 11    1023  1302   6  4  9 15
2121  3201  12  7  9  6    2031  1203  10  3 13  8    2301  3102  12 14  1  7

180/   95/         868#     181/  502/         869#     184/  179/         871#
1032  0231   5  3 16 10    1032  2130   7  2 16  9    1032  0330   5  4 16  9
1203  1032   6  9  4 15    1302  1203   6 15  1 12    0213  1221   2 11  7 14
2130  2301  11  8 13  2    2301  2130  11 14  4  5    3120  2112  15  6 10  3
2301  3102  12 14  1  7    2031  1203  10  3 13  8    2301  3003  12 13  1  8

185/  135/         875#     186/   56/         876#     187/   95/         880#
1032  0312   5  4 14 11    1032  0132   5  2 16 11    1032  0231   5  3 16 10
```

```
 0303  1023   2 13  3 16    0303  3201   4 15  1 14    0312  1032   2 13  8 11
 3120  2310  15  8 10  1    3210  0132  13 10  8  3    3021  2301  15  4  9  6
 2211  3021  12  9  7  6    2121  3201  12  7  9  6    2301  3102  12 14  1  7
[Solution Counts = 880]
```

Let me modify the list, with sorting the data and referring to the old solutions.

```
** Smart List of Standard MS44 composed by New Euler's Method' **
   (Unit_Nmb: High/ Low/; Solution_Nmb: [New] Old# Sum_Checks\/||)
 52/H 128/L  [  1]    1#\34/34|     50/H  89/L  [ 26]   26#\34/34|
0330  0303    1 16 13  4|34|34|    0330  0213    1 15 14  4|34|34|
2211  3030   12  9  8  5|34|34|    2112  3120   12  6  7  9|34|34|
1122  1212    6  7 10 11|34|34|    1221  3120    8 10 11  5|34|34|
3003  2121   15  2  3 14|34|34|    3003  0213   13  3  2 16|34|34|

 50/H  43/L  [ 62]   62#\34/34|     50/H  13/L  [ 94]   95#\34/34|
0330  0123    1 14 15  4|34|34|    0330  0033    1 13 16  4|34|34|
2112  3210   12  7  6  9|34|34|    2112  3300   12  8  5  9|34|34|
1221  3210    8 11 10  5|34|34|    1221  1122    6 10 11  7|34|34|
3003  0123   13  2  3 16|34|34|    3003  2211   15  3  2 14|34|34|

 72/H 155/L  [115]  116#\34/34|     70/H  98/L  [147]  148#\34/34|
0231  0321    1 12 15  6|34|34|    0231  0231    1 11 16  6|34|34|
3201  1032   14  9  4  7|34|34|    3102  1320   14  8  3  9|34|34|
0132  2301    3  8 13 10|34|34|    0231  3102    4 10 13  7|34|34|
3102  3012   16  5  2 11|34|34|    3102  2013   15  5  2 12|34|34|

 70/H  54/L  [158]  159#\34/34|     73/H  11/L  [172]  172#\34/34|
0231  0132    1 10 16  7|34|34|    0231  0033    1  9 16  8|34|34|
3102  2310   15  8  2  9|34|34|    3210  2301   15 12  5  2|34|34|
0231  3201    4 11 13  6|34|34|    0123  3210    4  7 10 13|34|34|
3102  1023   14  5  3 12|34|34|    3102  1122   14  6  3 11|34|34|

106/H 154/L  [177]  177#\34/34|    106/H  98/L  [194]  194#\34/34|
0132  0321    1  8 15 10|34|34|    0132  0231    1  7 16 10|34|34|
3201  0321   13 12  3  6|34|34|    3201  1320   14 12  3  5|34|34|
0132  3012    4  5 14 11|34|34|    0132  3102    4  6 13 11|34|34|
3201  3012   16  9  2  7|34|34|    3201  2013   15  9  2  8|34|34|

106/H  54/L  [203]  203#\34/34|    113/H   4/L  [207]  207#\34/34|
0132  0132    1  6 16 11|34|34|    0132  0033    1  5 16 12|34|34|
3201  2310   15 12  2  5|34|34|    2301  1122   10 14  3  7|34|34|
0132  3201    4  7 13 10|34|34|    1032  3300    8  4 13  9|34|34|
3201  1023   14  9  3  8|34|34|    3201  2211   15 11  2  6|34|34|

 50/H 354/L  [209]  209#\34/34|     50/H 292/L  [241]  241#\34/34|
0330  1302    2 16 13  3|34|34|    0330  1212    2 15 14  3|34|34|
2112  3210   12  7  6  9|34|34|    2112  3030   12  5  8  9|34|34|
1221  0123    5 10 11  8|34|34|    1221  2121    7 10 11  6|34|34|
3003  2031   15  1  4 14|34|34|    3003  0303   13  4  1 16|34|34|

 50/H 247/L  [266]  267#\34/34|     50/H 233/L  [289]  289#\34/34|
0330  1122    2 14 15  3|34|34|    0330  1032    2 13 16  3|34|34|
2112  2211   11  7  6 10|34|34|    2112  3210   12  7  6  9|34|34|
1221  0033    5  9 12  8|34|34|    1221  0123    5 10 11  8|34|34|
3003  3300   16  4  1 13|34|34|    3003  2301   15  4  1 14|34|34|

 70/H 361/L  [311]  312#\34/34|     70/H 312/L  [320]  320#\34/34|
0231  1320    2 12 15  5|34|34|    0231  1230    2 11 16  5|34|34|
3102  0231   13  7  4 10|34|34|    3102  1230   14  7  4  9|34|34|
0231  2013    3  9 14  8|34|34|    0231  2103    3 10 13  8|34|34|
3102  3102   16  6  1 11|34|34|    3102  2103   15  6  1 12|34|34|

 73/H 254/L  [354]  354#\34/34|     70/H 228/L  [359]  362#\34/34|
```

```
 0231  1122    2 10 15  7|34|34|    0231  1032    2  9 16  7|34|34|
 3210  3210   16 11  6  1|34|34|    3102  2301   15  8  1 10|34|34|
 0123  2301    3  8  9 14|34|34|    0231  2301    3 12 13  6|34|34|
 3102  0033   13  5  4 12|34|34|    3102  1032   14  5  4 11|34|34|
106/H 361/L  [373]  373#\34/34|   106/H 312/L  [382]  382#\34/34|
 0132  1320    2  8 15  9|34|34|    0132  1230    2  7 16  9|34|34|
 3201  0231   13 11  4  6|34|34|    3201  1230   14 11  4  5|34|34|
 0132  2013    3  5 14 12|34|34|    0132  2103    3  6 13 12|34|34|
 3201  3102   16 10  1  7|34|34|    3201  2103   15 10  1  8|34|34|
113/H 241/L  [399]  399#\34/34|   106/H 228/L  [401]  402#\34/34|
 0132  1122    2  6 15 11|34|34|    0132  1032    2  5 16 11|34|34|
 2301  0033    9 13  4  8|34|34|    3201  2301   15 12  1  6|34|34|
 1032  2211    7  3 14 10|34|34|    0132  2301    3  8 13 10|34|34|
 3201  3300   16 12  1  5|34|34|    3201  1032   14  9  4  7|34|34|
 22/H 570/L  [409]  409#\34/34|    22/H 754/L  [575]  577#\34/34|
 0321  2310    3 16 10  5|34|34|    0321  3300    4 16  9  5|34|34|
 3102  2130   15  6  4  9|34|34|    3102  0123   13  6  3 12|34|34|
 0231  1203    2 11 13  8|34|34|    0231  1212    2 11 14  7|34|34|
 3012  1023   14  1  7 12|34|34|    3012  2031   15  1  8 10|34|34|
129/H 184/L  [753]  753#\34/34|   141/H 361/L  [817]  817#\34/34|
 1302  0330    5 16  4  9|34|34|    1302  1320    6 16  3  9|34|34|
 3210  2112   15 10  6  3|34|34|    0213  0231    1 11  8 14|34|34|
 0123  1221    2  7 11 14|34|34|    3120  2013   15  5 10  4|34|34|
 2031  3003   12  1 13  8|34|34|    2031  3102   12  2 13  7|34|34|
131/H 571/L  [865]  865#\34/34|
 1302  2310    7 16  2  9|34|34|
 2031  2310   11  4 14  5|34|34|
 1032  1023    6  1 15 12|34|34|
 2301  1023   10 13  3  8|34|34|
[Solution Counts(New/Old) = 880/880]  [OK!]
```

11. They prove we have really succeeded in reconstructing all the 880 solutions of Standard MS44 by the 4-th Increment System. We could neither make any other type of solutions, nor miss any one of necessary solutions at all.

Don't you think the whole process implies the inner structure of our Standard type of MS44 very clearly? I admit it is a little too complex with some exceptional irregularities, but the complexity itself is the nature of our object, isn't it?

Thus, we have succeeded in reconstructing all the solutions of 3 typical MS44 by the 4-th Increment System and our New Euler's Method. We could have fulfilled half of the dream of Euler School, I believe, without using the old conventional 'Greco-Latinian Method' they have loved for a long time.

What was the **Greco-Latinian Method**, then?

It was really a beautiful method to produce every kind of magic things. We have used only (0, 1, 2 and 3) to compose any rows, columns and diagonals of each Euler Unit to fulfill the constant sum 6: `0+1+2+3=6;`

But we have not allowed the patterns like `0+0+3+3=6;` or `1+2+1+2=6;` there. Any number of (0, 1, 2 and 3) must come strictly once in each line. The queuing order is free, but the contents of essential parts must obey the 'Latin Condition'.

This method is quite effective for composing the Pandiagonal Magic Squares of Order 5 and the Simultaneous type of MS55: Self-complementary and Pandiagonal. It can perfectly compose all the solutions of those types. How impressive it is!

But it is not very effective for the other types of magic things, especially of the

even orders. We can compose only a few solutions of those types, not all of them. We cannot help using the Binary System instead, or of the other bases there.

Why have they obeyed such the conventional method as this for a long time?

I imagine they have obeyed for their aesthetic reason. For it is very beautiful. Everyone wants everything beautiful and chooses to obey the beautiful Latin Condition to make everything.

But the facts before us are not always beautiful, and the theories about them are not always beautiful, too. We must always face the facts honestly, I would say.

12. Let me explain a little about the 'third rule' here.

We have always accepted the **third rule** whenever we should compose every Euler Units: Any number of (0, 1, 2 and 3) must come just 4 times in each units equally as often as the others do.

What does it mean by this condition? Why do we always have to accept it?

I think this condition directly corresponds to the basic rule that we must use every number of (1, 2, 3, 4, ...,14,15,16) strictly once to compose our magic squares of order 4, and must not un-use any one at all. Any duplication or drop-out of numbers is strictly prohibited. This is one of the oldest 'promises' we must always obey for a long time.

First of all, let me rewrite this rule into the mathematical notation as follows:

> "We must use any integer of (0, 1, 2, 3, ...,13,14,15) strictly once and must not un-use any one at all."

And then let me rewrite it by using the notation of 4-th Increment System:

> "We must use any integer of (00,01,02,03, 10,11,12,13, 20,21,22,23, 30,31,32,33(N4i)) strictly once and must not un-use any one at all."

Why don't you decompose this rule into Euler Units of high position and low?

Classical/				Mathematical/				Mathem/N4i ->				EUnit: H/				L/			
1	2	3	4	0	1	2	3	00	01	02	03	0	0	0	0	0	1	2	3
5	6	7	8	4	5	6	7	10	11	12	13	1	1	1	1	0	1	2	3
9	10	11	12	8	9	10	11	20	21	22	23	2	2	2	2	0	1	2	3
13	14	15	16	12	13	14	15	30	31	32	33	3	3	3	3	0	1	2	3

> "We must use any number of (0, 1, 2 and 3) just 4 times equally as often as the other numbers to compose each Euler Unit."

Do you understand one of the oldest promises certainly directs our Euler Units in such the way as the 'third rule'? How interesting it is!

13. Let me append one more note about "Euler Squares" and "Non-Euler Types".

In the former article I have reported about the 528 solutions of "Euler Squares" for the standard MS44 made by the Binary System. In each Euler Unit of them any row, any column and any primary diagonal of them always adds up to the same sum 2 without any exceptions. It means that you can count the "Non-Euler Type" among the 880 standard solutions as many as 352(=880-528) pieces.

But in this section I have made another type of 656 **Euler Squares** for the same object by the 4-th Increment System. Any row, column or primary diagonal in each unit of them always adds up to the constant sum 6. It means you can have the **Non-Euler Type** as many as 224(=880-656) pieces, doesn't it?

Why do we have to have two kinds of Non-Euler Type with different counts for

the same object? Is there anything wrong? What mistake did I make?

But I am truly self-confident I am right. It must be thought as a typical example illustrating the fact that any different ways often make the different results.

Let's get back to our starting point, again to the solution list of our object with decomposed units by the 4-th Increment and Binary Number Systems.

```
** Standard Magic Squares of Order 4 Composed by the Ordinary Method **
[Solution#(Classical) /Mathematical with Sum Checks: \d1/d2|rw|cl| /D4i /D2i]
          1#   /Math      \30/30|   /D4i: Irregular     /D2i: Irregular
 1 16 13  4    0 15 12  3|30|30|   0 3 3 0   0 3 0 3   0110 0110 0101 0101
12  9  8  5   11  8  7  4|30|30|   2 2 1 1   3 0 3 0   1100 0011 1010 1010
 6  7 10 11    5  6  9 10|30|30|   1 1 2 2   1 2 1 2   0011 1100 0101 1010
15  2  3 14   14  1  2 13|30|30|   3 0 0 3   2 1 2 1   1001 1001 1010 0101

          7#   /Math      \30/30|   /D4i               /D2i
 1 16 11  6    0 15 10  5|30|30|   0 3 2 1   0 3 2 1   0110 0101 0110 0101
13  4  7 10   12  3  6  9|30|30|   3 0 1 2   0 3 2 1   1001 1010 0110 0101
 8  9 14  3    7  8 13  2|30|30|   1 2 3 0   3 0 1 2   0110 1010 1001 1010
12  5  2 15   11  4  1 14|30|30|   2 1 0 3   3 0 1 2   1001 0101 1001 1010

         11#   /Math      \30/30|   /D4i               /D2i
 1 16 10  7    0 15  9  6|30|30|   0 3 2 1   0 3 1 2   0110 0101 0101 0110
13  4  6 11   12  3  5 10|30|30|   3 0 1 2   0 3 1 2   1001 1010 0101 0110
 8  9 15  2    7  8 14  1|30|30|   1 2 3 0   3 0 2 1   0110 1010 1010 1001
12  5  3 14   11  4  2 13|30|30|   2 1 0 3   3 0 2 1   1001 0101 1010 1001

         15#   /Math      \30/30|   /D4i               /D2i: Irregular
 1 16  9  8    0 15  8  7|30|30|   0 3 2 1   0 3 0 3   0110 0101 0101 0101
15 10  7  2   14  9  6  1|30|30|   3 2 1 0   2 1 2 1   1100 1010 1010 0101
 4  5 12 13    3  4 11 12|30|30|   0 1 2 3   3 0 3 0   0011 0101 1010 1010
14  3  6 11   13  2  5 10|30|30|   3 0 1 2   1 2 1 2   1001 1010 0101 1010

         20#   /Math      \30/30|   /D4i               /D2i
 1 16  7 10    0 15  6  9|30|30|   0 3 1 2   0 3 2 1   0101 0110 0110 0101
 6 11  4 13    5 10  3 12|30|30|   1 2 0 3   1 2 3 0   0101 1001 0110 1010
12  5 14  3   11  4 13  2|30|30|   2 1 3 0   3 0 1 2   1010 0110 1001 1010
15  2  9  8   14  1  8  7|30|30|   3 0 2 1   2 1 0 3   1010 1001 1001 0101

         22#   /Math      \30/30|   /D4i               /D2i
 1 16  6 11    0 15  5 10|30|30|   0 3 1 2   0 3 1 2   0101 0110 0101 0110
 7 10  4 13    6  9  3 12|30|30|   1 2 0 3   2 1 3 0   0101 1001 1010 0110
12  5 15  2   11  4 14  1|30|30|   2 1 3 0   3 0 2 1   1010 0110 1010 1001
14  3  9  8   13  2  8  7|30|30|   3 0 2 1   1 2 0 3   1010 1001 0101 1001

         24#   /Math      \30/30|   /D4i: Irregular     /D2i: Irregular
 1 16  5 12    0 15  4 11|30|30|   0 3 1 2   0 3 0 3   0101 0110 0101 0101
10 13  4  7    9 12  3  6|30|30|   2 3 0 1   1 0 3 2   1100 0101 0011 1010
 8  3 14  9    7  2 13  8|30|30|   1 0 3 2   3 2 1 0   0011 1010 1100 1010
15  2 11  6   14  1 10  5|30|30|   3 0 2 1   2 1 2 1   1010 1001 1010 0101

         26#   /Math      \30/30|   /D4i               /D2i
 1 15 14  4    0 14 13  3|30|30|   0 3 3 0   0 2 1 3   0110 0110 0101 0011
12  6  7  9   11  5  6  8|30|30|   2 1 1 2   3 1 2 0   1001 0110 1010 1100
 8 10 11  5    7  9 10  4|30|30|   1 2 2 1   3 1 2 0   0110 1001 1010 1100
13  3  2 16   12  2  1 15|30|30|   3 0 0 3   0 2 1 3   1001 1001 0101 0011

         36#   /Math      \30/30|   /D4i               /D2i
 1 15 12  6    0 14 11  5|30|30|   0 3 2 1   0 2 3 1   0110 0101 0110 0011
14  4  7  9   13  3  6  8|30|30|   3 0 1 2   1 3 2 0   1001 1010 0110 1100
 8 10 13  3    7  9 12  2|30|30|   1 2 3 0   3 1 0 2   0110 1010 1001 1100
11  5  2 16   10  4  1 15|30|30|   2 1 0 3   2 0 1 3   1001 0101 1001 0011

         46#   /Math      \30/30|   /D4i: Irregular     /D2i: Irregular
 1 15 10  8    0 14  9  7|30|30|   0 3 2 1   0 2 1 3   0110 0101 0101 0011
16  6  3  9   15  5  2  8|30|30|   3 1 0 2   3 1 2 0   1001 1100 1010 1100
 5 11 14  4    4 10 13  3|30|30|   1 2 3 0   0 2 1 3   0110 1010 0101 0011
12  2  7 13   11  1  6 12|30|30|   2 0 1 3   3 1 2 0   1001 0011 1010 1100
```

```
        56#   /Math       \30/30|   /D4i                 /D2i
 1 15  8 10    0 14  7  9|30|30|   0 3 1 2   0 2 3 1   0101 0110 0110 0011
13 12  3  6   12 11  2  5|30|30|   3 2 0 1   0 3 2 1   1100 1001 0110 0101
 4  5 14 11    3  4 13 10|30|30|   0 1 3 2   3 0 1 2   0011 0110 1001 1010
16  2  9  7   15  1  8  6|30|30|   3 0 2 1   3 1 0 2   1010 1001 1001 1100

        60#   /Math       \30/30|   /D4i                 /D2i
 1 15  6 12    0 14  5 11|30|30|   0 3 1 2   0 2 1 3   0101 0110 0101 0011
 8 10  3 13    7  9  2 12|30|30|   1 2 0 3   3 1 2 0   0101 1001 1010 1100
11  5 16  2   10  4 15  1|30|30|   2 1 3 0   2 0 3 1   1010 0110 1010 0011
14  4  9  7   13  3  8  6|30|30|   3 0 2 1   1 3 0 2   1010 1001 0101 1100

        62#   /Math       \30/30|   /D4i                 /D2i
 1 14 15  4    0 13 14  3|30|30|   0 3 3 0   0 1 2 3   0110 0110 0011 0101
12  7  6  9   11  6  5  8|30|30|   2 1 1 2   3 2 1 0   1001 0110 1100 1010
 8 11 10  5    7 10  9  4|30|30|   1 2 2 1   3 2 1 0   0110 1001 1100 1010
13  2  3 16   12  1  2 15|30|30|   3 0 0 3   0 1 2 3   1001 1001 0011 0101

        72#   /Math       \30/30|   /D4i                 /D2i
 1 14 12  7    0 13 11  6|30|30|   0 3 2 1   0 1 3 2   0110 0101 0011 0110
15  4  6  9   14  3  5  8|30|30|   3 0 1 2   2 3 1 0   1001 1010 1100 0110
 8 11 13  2    7 10 12  1|30|30|   1 2 3 0   3 2 0 1   0110 1010 1100 1001
10  5  3 16    9  4  2 15|30|30|   2 1 0 3   1 0 2 3   1001 0101 0011 1001

        78#   /Math       \30/30|   /D4i: Irregular     /D2i: Irregular
 1 14 11  8    0 13 10  7|30|30|   0 3 2 1   0 1 2 3   0110 0101 0011 0101
16  5  4  9   15  4  3  8|30|30|   3 1 0 2   3 0 3 0   1001 1100 1010 1010
 7 12 13  2    6 11 12  1|30|30|   1 2 3 0   2 3 0 1   0110 1010 1100 0101
10  3  6 15    9  2  5 14|30|30|   2 0 1 3   1 2 1 2   1001 0011 0101 1010

        84#   /Math       \30/30|   /D4i                 /D2i
 1 14  8 11    0 13  7 10|30|30|   0 3 1 2   0 1 3 2   0101 0110 0011 0110
13 12  2  7   12 11  1  6|30|30|   3 2 0 1   0 3 1 2   1100 1001 0101 0110
 4  5 15 10    3  4 14  9|30|30|   0 1 3 2   3 0 2 1   0011 0110 1010 1001
16  3  9  6   15  2  8  5|30|30|   3 0 2 1   3 2 0 1   1010 1001 1100 1001

        88#   /Math       \30/30|   /D4i: Irregular     /D2i: Irregular
 1 14  7 12    0 13  6 11|30|30|   0 3 1 2   0 1 2 3   0101 0110 0011 0101
11 13  2  8   10 12  1  7|30|30|   2 3 0 1   2 0 1 3   1100 0101 1001 0011
 6  4 15  9    5  3 14  8|30|30|   1 0 3 2   1 3 2 0   0011 1010 0110 1100
16  3 10  5   15  2  9  4|30|30|   3 0 2 1   3 2 1 0   1010 1001 1100 1010

        94#   /Math       \30/30|   /D4i                 /D2i
 1 13 16  4    0 12 15  3|30|30|   0 3 3 0   0 0 3 3   0110 0110 0011 0011
12  8  5  9   11  7  4  8|30|30|   2 1 1 2   3 3 0 0   1001 0110 1100 1100
 7 11 10  6    6 10  9  5|30|30|   1 2 2 1   2 2 1 1   0110 1001 1100 0011
14  2  3 15   13  1  2 14|30|30|   3 0 0 3   1 1 2 2   1001 1001 0011 1100

       100#   /Math       \30/30|   /D4i                 /D2i
 1 13 12  8    0 12 11  7|30|30|   0 3 2 1   0 0 3 3   0110 0101 0011 0011
16  4  5  9   15  3  4  8|30|30|   3 0 1 2   3 3 0 0   1001 1010 1100 1100
 7 11 14  2    6 10 13  1|30|30|   1 2 3 0   2 2 1 1   0110 1010 1100 0011
10  6  3 15    9  5  2 14|30|30|   2 1 0 3   1 1 2 2   1001 0101 0011 1100

       111#   /Math       \30/30|   /D4i                 /D2i: Irregular
 1 13  8 12    0 12  7 11|30|30|   0 3 1 2   0 0 3 3   0101 0110 0011 0011
16 11  2  5   15 10  1  4|30|30|   3 2 0 1   3 2 1 0   1100 1001 1100 1010
 3  6 15 10    2  5 14  9|30|30|   0 1 3 2   2 1 2 1   0011 0110 1010 0101
14  4  9  7   13  3  8  6|30|30|   3 0 2 1   1 3 0 2   1010 1001 0101 1100

       115#   /Math       \30/30|   /D4i                 /D2i
 1 12 15  6    0 11 14  5|30|30|   0 2 3 1   0 3 2 1   0110 0011 0110 0101
14  7  4  9   13  6  3  8|30|30|   3 1 0 2   1 2 3 0   1001 1100 0110 1010
 8 13 10  3    7 12  9  2|30|30|   1 3 2 0   3 0 1 2   0110 1100 1001 1010
11  2  5 16   10  1  4 15|30|30|   2 0 1 3   2 1 0 3   1001 0011 1001 0101

       125#   /Math       \30/30|   /D4i                 /D2i
 1 12 14  7    0 11 13  6|30|30|   0 2 3 1   0 3 1 2   0110 0011 0101 0110
15  6  4  9   14  5  3  8|30|30|   3 1 0 2   2 1 3 0   1001 1100 1010 0110
 8 13 11  2    7 12 10  1|30|30|   1 3 2 0   3 0 2 1   0110 1100 1010 1001
10  3  5 16    9  2  4 15|30|30|   2 0 1 3   1 2 0 3   1001 0011 0101 1001
```

```
        131#   /Math      \30/30|   /D4i              /D2i: Irregular
 1 12 13  8    0 11 12  7|30|30|   0 2 3 1   0 3 0 3   0110 0011 0101 0101
16  7  2  9   15  6  1  8|30|30|   3 1 0 2   3 2 1 0   1001 1100 1100 1010
 3 10 15  6    2  9 14  5|30|30|   0 2 3 1   2 1 2 1   0110 0011 1010 0101
14  5  4 11   13  4  3 10|30|30|   3 1 0 2   1 0 3 2   1001 1100 0011 1010

        147#   /Math      \30/30|   /D4i              /D2i
 1 11 16  6    0 10 15  5|30|30|   0 2 3 1   0 2 3 1   0110 0011 0110 0011
14  8  3  9   13  7  2  8|30|30|   3 1 0 2   1 3 2 0   1001 1100 0110 1100
 7 13 10  4    6 12  9  3|30|30|   1 3 2 0   2 0 1 3   0110 1100 1001 0011
12  2  5 15   11  1  4 14|30|30|   2 0 1 3   3 1 0 2   1001 0011 1001 1100

        158#   /Math      \30/30|   /D4i              /D2i
 1 10 16  7    0  9 15  6|30|30|   0 2 3 1   0 1 3 2   0110 0011 0011 0110
15  8  2  9   14  7  1  8|30|30|   3 1 0 2   2 3 1 0   1001 1100 1100 0110
 6 13 11  4    5 12 10  3|30|30|   1 3 2 0   1 0 2 3   0110 1100 0011 1001
12  3  5 14   11  2  4 13|30|30|   2 0 1 3   3 2 0 1   1001 0011 1100 1001

        172#   /Math      \30/30|   /D4i              /D2i: Irregular
 1  9 16  8    0  8 15  7|30|30|   0 2 3 1   0 0 3 3   0110 0011 0011 0011
15 12  5  2   14 11  4  1|30|30|   3 2 1 0   2 3 0 1   1100 1010 1100 0101
 4  7 10 13    3  6  9 12|30|30|   0 1 2 3   3 2 1 0   0011 0101 1100 1010
14  6  3 11   13  5  2 10|30|30|   3 1 0 2   1 1 2 2   1001 1100 0011 1100

        177#   /Math      \30/30|   /D4i              /D2i
 1  8 15 10    0  7 14  9|30|30|   0 1 3 2   0 3 2 1   0011 0110 0110 0101
13 12  3  6   12 11  2  5|30|30|   3 2 0 1   0 3 2 1   1100 1001 0110 0101
 4  5 14 11    3  4 13 10|30|30|   0 1 3 2   3 0 1 2   0011 0110 1001 1010
16  9  2  7   15  8  1  6|30|30|   3 2 0 1   3 0 1 2   1100 1001 1001 1010

        194#   /Math      \30/30|   /D4i              /D2i
 1  7 16 10    0  6 15  9|30|30|   0 1 3 2   0 2 3 1   0011 0110 0110 0011
14 12  3  5   13 11  2  4|30|30|   3 2 0 1   1 3 2 0   1100 1001 0110 1100
 4  6 13 11    3  5 12 10|30|30|   0 1 3 2   3 1 0 2   0011 0110 1001 1100
15  9  2  8   14  8  1  7|30|30|   3 2 0 1   2 0 1 3   1100 1001 1001 0011

        203#   /Math      \30/30|   /D4i              /D2i
 1  6 16 11    0  5 15 10|30|30|   0 1 3 2   0 1 3 2   0011 0110 0011 0110
15 12  2  5   14 11  1  4|30|30|   3 2 0 1   2 3 1 0   1100 1001 1100 0110
 4  7 13 10    3  6 12  9|30|30|   0 1 3 2   3 2 0 1   0011 0110 1100 1001
14  9  3  8   13  8  2  7|30|30|   3 2 0 1   1 0 2 3   1100 1001 0011 1001

        207#   /Math      \30/30|   /D4i: Irregular   /D2i: Irregular
 1  5 16 12    0  4 15 11|30|30|   0 1 3 2   0 0 3 3   0011 0110 0011 0011
10 14  3  7    9 13  2  6|30|30|   2 3 0 1   1 1 2 2   1100 0101 0011 1100
 8  4 13  9    7  3 12  8|30|30|   1 0 3 2   3 3 0 0   0011 1010 1100 1100
15 11  2  6   14 10  1  5|30|30|   3 2 0 1   2 2 1 1   1100 1001 1100 0011
. . . . .
```

Watch the above list carefully, please.

I colored all the irregular diagonals there in yellow. They don't add up to the constant sums: 6 in the 4-th Increment units and 2 in the binary ones.

All diagonals colored green or with no color add up to the regular sums.

Take the solution 1# for instance. It is so irregular both in the units /D4i and in the units /D2i that it must be certainly classified into the **Non-Euler Type**.

In the same way the solutions 24#, 46#, 78# and so on are so irregular that they must be taken into the same category Non-Euler Type.

But watch the solutions 15# and 111# above carefully, and find they are regular enough to be classified into the **Euler Type** by the 4-th Increment Number System, while in the Binary Units they are too irregular to be taken in the same category.

The solutions 131# and 172# belong to the same type, since they are regular enough to be classified into the set of Euler Squares by the 4-th Increment System, while in the Binary Units they are too irregular to be taken into the same set.

Such the mysterious solutions as these really exist in our object Standard MS44.

And the existence of this kind is the true reason why we should have two different sets of Euler Squares with different counts and inevitably have two different Non-Euler Types with different counts.

Now we must always mention about which method we used to compose and count our objects, whether by the 4-th Increment System or by the Binary System.

Let me re-define the name “Euler Squares” and “Non-Euler Type” here.

We call “**Euler Squares**” for those types whose rows, columns and diagonals in each unit add up to the same constant sum. They must prove the structural similarity between their component units and the whole objects themselves.

And we now call “**Non-Euler Type**” for those objects whose similarity is broken in any way between their component units and the whole squares themselves.

We do not care about the content patterns of any row, column or diagonal in each unit. 0+0+3+3=6; 1+1+1+3=6 or 1+1+2+2=6 are always all right enough to be accepted, on top of the usual case 0+1+2+3=6.

If you want to have the only content pattern (0, 1, 2, 3) for those essential parts in each unit, you may continue to call it as “**Latin Unit**” and call the whole object “**Greco-Latin Square**” instead of “Euler Square” as they have long called.

You may also call the “**Complete** Euler Squares” for those Greco-Latin Squares.

Let’s differentiate the meanings between those two types: **Greco-Latin Squares** (or **Complete** Euler Squares) and simple **Euler Squares** now, though they have long been taken as the same thing.

(Kanji Setsuda’s New Definition in 2012)

(The original Japanese version was written on Dec.18, 2011;
English version was written on Feb. 5, 2012;
Revised on Mar. 6, 2012 with MacOSX 10.7.3 & Xcode 4.2.1)

---

by **Kanji Setsuda**: E-Mail Address:<jag12001@nifty.com>
http://homepage2.nifty.com/KanjiSetsuda/pages/EnglishP1.html