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

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

Section 3-1: How to make 'Complete Euler Squares' of Order 5

#1. What is the 'Complete Euler Square' like?

It is sometimes called "Greco-Latin Square". They say basic idea of that type was mentioned first by Legendary Leonhard Euler(1707-1783). What does it look like?

First of all we have to change our number system from classical one into modern.

You know about the so-called 'Positional Number System of Base N', don't you?

** Basic Form and our Object Square in Classical Notation **

[Basic Form]

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

[Pan-diagonal]

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

** Pan-diagonal Magic Square 5x5 in the Modern Notation **

[Pan-diagonal]

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

[Modern Notation]

[Decomposed by PNS of Base 5]

0	4	3	2	1	0	4	3	2	1
2	1	0	4	3	2	1	0	4	3
4	3	2	1	0	4	3	2	1	0
1	0	4	3	2	1	0	4	3	2
3	2	1	0	4	3	2	1	0	4
0	4	3	2	1	0	4	3	2	1
2	1	0	4	3	2	1	0	4	3

[High]

2	4	1	3	0	2	4	1	3	0
1	3	0	2	4	1	3	0	2	4
0	2	4	1	3	0	2	4	1	3
4	1	3	0	2	4	1	3	0	2
3	0	2	4	1	3	0	2	4	1
2	4	1	3	0	2	4	1	3	0
1	3	0	2	4	1	3	0	2	4

[Low]

Let me take a system of the Base 5 for instance. Watch and study the figures above, especially the last two decomposed patterns for 'Complete Euler Square' 5x5.

Each positional layer for high values or low ones is drawn separately as shown above. Any value of 25 numbers is divided by 5. The integer part of quotient is listed in the high layer and the remedy modulo(5) is put in the low one.

According to this system each sum of every column, of every row and even of every pan-diagonal in this example is always calculated by such the same form of equation:

(0+1+2+3+4)x5+(0+1+2+3+4)=10x5+10=60(Decimal)

The equal sum means the Magic Constant.

Yes. It is certainly an example solution of 'Complete Euler Square' 5x5.

The constant sum 60 of this example is equivalent to 65 in our classical notation, since modern system uses integers: 0~24, instead of natural numbers: 1~25.

#2. Unique Properties of these 'Complete Euler Squares'

Each 'Complete Euler Square' has such the beautiful properties as follows:

(1) In each layer every column, every row and every pan-diagonal consists of {0, 1, 2, 3 and 4} and it has neither repetition, nor drop-off of any number.

(2) Any value of {0, 1, 2, 3 or 4} appears always five times in each layer.

(3) The value combination of high layer and low one in any location is logically the same with the value of the same location in the figure on the left hand side.

34(N5i)= 3x5 +4(=15+4=19(Decimal): equivalent to 20 in the Classical Notation)

Neither repetition nor drop-off of a certain combination should be found.

That means any of all integers 0~24 must be used strictly once to make our object.

Check and find that it has these beautiful properties in the example put above.

They say Euler did not mention about pan-diagonals, but I know all pan-diagonals also have this property(1), and I want to call those which has all the miraculous properties above for "**Complete Euler Squares**" from now on.

#3. How to construct such 'Complete Euler Squares'

Can we make such a beautiful square as 'Complete Euler Square' directly?

Why don't you try to dictate our program simply expressing the idea literally after these beautiful definitions themselves?

[1] We usually want the variable N to take any value of 0~24 as dictated below.

But why don't you use two variables `N_high and N_low`, each of which takes any value of {0, 1, 2, 3 or 4}? Each variable is expected to stand for any value of its own positional layer.

How about dictating such a number generator program with double loops of `for(...){...}` sentences as follows?

```
//* Old *
  for(n=0;n<25;n++){
    nm[1]=n;
    // ...;
  }
//
```

```
//* New *
  for(a=0;a<5;a++){      // a==d/5: for high
    for(b=0;b<5;b++){    // b==d%5: for low
      d=a*5+b;           // 'd' stands for N itself
      n[1]=d;
      // ...;
    }
  }
//
```

[2] How do we realize that we must take any value of {0, 1, 2, 3 or 4} strictly once and must not use it twice or more often in any row, any column and any pan-diagonal of our object?

We have usually used the so-called "`used_flag`" for this purpose.

When you use the value N, you should set the `used_flag` for it as '`true`', meaning '`used`'. And when you finish using N, you should reset the flag as '`false`', meaning '`not-used`', and may return to the former step.

Whenever you make a new job, you should always check if the `used_flag` is '`true`' or '`false`' before all.

```
//* Old *
  for(n=0;n<25;n++){
    if(uflg[n]==0){    // Check if the value N is used or not
      nm[3]=n;         // If it is not used, then ...
      uflg[n]=1;       // Set Used_flag for the value N
      nextproc();      // Go to the next procedure
      uflg[n]=0;       // Reset Used_flag for the value N
```

```
    }
  }
//
```

Why don't you remake it into the new style below to meet our double structure?

```
//* New *
  for(a=0;a<5;a++){
    if(uflg_high[a]==0){   // Check if it is used or not
   for(b=0;b<5;b++){
     if(uflg_low[b]==0){   // Check if it is used or not
      n=a*5+b; nm[3]=n;    // Produce N from a and b ...
      if(uflg[n]==0){      // Check if the value N is used or not
       uflg[n]=1;          // Set Used_flag for the value N
       uflg_high[a]=1;     // Set Used_flag_high for Value a
       uflg_low[b]=1;      // Set Used_flag_low for Value b
       nextproc();         // Go to the next procedure
       uflg_high[a]=0;     // Reset Used_flag_high for Value a
       uflg_low[b]=0;      // Reset Used_flag_low for Value b
       uflg[n]=0;          // Reset Used_flag for Value N
      }
   }
    }
  }
//
```

[3] How many sets of used_flags in all do we have to prepare, then?
We have to prepare 10 sets of used_flags for 5 columns high and low, 10 for 5 rows high and low, and 20 for 10 pan-diagonals high and low:
We need 40 sets in all!

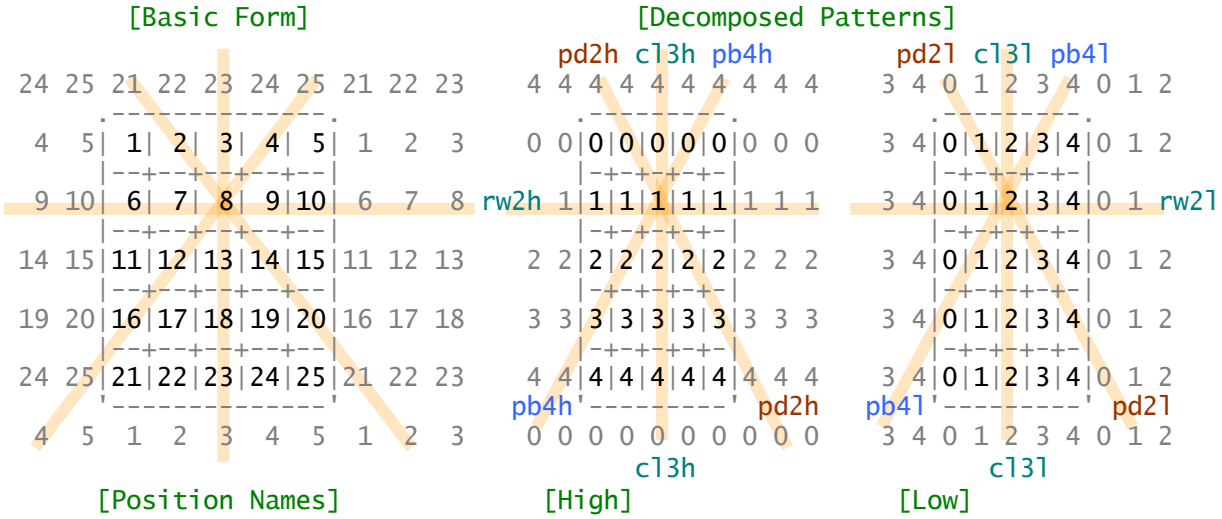
```
**  Basic Forms for Pan-diagonal Magic Squares of Order 5  **

 24 25 21 22 23 24 25 21 22        pd5 pd1  pd3    pd5 pb4 pb5
     .--------------.                24 25 21 22 23 24 25 21 22
 4  5| 1| 2| 3| 4| 5| 1  2rw1    pd4    .--------------. pb1
     |--+--+--+--+--|              4  5| 1| 2| 3| 4| 5| 1  2
 9 10| 6| 7| 8| 9|10| 6  7rw2    pd3   |--+--+--+--+--| pb2
     |--+--+--+--+--|              9 10| 6| 7| 8| 9|10| 6  7
14 15|11|12|13|14|15|11 12rw3    pd2pb1|--+--+--+--+--| pb3
     |--+--+--+--+--|             14 15|11|12|13|14|15|11 12
19 20|16|17|18|19|20|16 17rw4      pb2 |--+--+--+--+--| pb4pd5
     |--+--+--+--+--|             19 20|16|17|18|19|20|16 17
24 25|21|22|23|24|25|21 22rw5      pb3 |--+--+--+--+--|    pd4
     '--------------'              24 25|21|22|23|24|25|21 22
 4  5  1  2  3  4  5  1  2         pb4 '--------------'    pd3
       cl2      cl5                4  5  1  2  3  4  5  1  2
                                    pb5     pd2   pd4      pd2

* Basic Conditions: C=60; *
  n1+n2+n3+n4+n5=C       ... rw1 | n1+n6+n11+n16+n21=C  ... cl1
  n6+n7+n8+n9+n10=C      ... rw2 | n2+n7+n12+n17+n22=C  ... cl2
  n11+n12+n13+n14+n15=C  ... rw3 | n3+n8+n13+n18+n23=C  ... cl3
  n16+n17+n18+n19+n20=C  ... rw4 | n4+n9+n14+n19+n24=C  ... cl4
  n21+n22+n23+n24+n25=C  ... rw5 | n5+n10+n15+n20+n25=C ... cl5
* Pan-diagonal Conditions: C=60; *
  n1+n7+n13+n19+n25=C    ... pd1 | n1+n10+n14+n18+n22=C ... pb1
  n2+n8+n14+n20+n21=C    ... pd2 | n2+n6+n15+n19+n23=C  ... pb2
  n3+n9+n15+n16+n22=C    ... pd3 | n3+n7+n11+n20+n24=C  ... pb3
  n4+n10+n11+n17+n23=C   ... pd4 | n4+n8+n12+n16+n25=C  ... pb4
  n5+n6+n12+n18+n24=C    ... pd5 | n5+n9+n13+n17+n21=C  ... pb5
```

We have to put their names each by each as shown above.

[4] Which set of used_flags should we check in one procedure when you want to take a certain value for any single variable?



Suppose if you want to take the value of N8 with 7(Decimal) for instance, you should check if the states of {cl3h[1], cl3l[2], rw2h[1], rw2l[2], pd2h[1], pd2l[2], pb4h[1], pb4l[2] and uflg[7]} are all 'false', meaning 'not-used'. You should watch and check 8 flags of 4 lines high and low and check one more flag for d itself.

#4. How to build our Computer Program for 'Complete Euler Square' 5x5

Let me skip my further explanation and come to my conclusion.

I will show you the list of my actual program here. It will tell you how I could make the 3600 Pan-diagonal Magic Squares by my new way of programming for 'Complete Euler Squares'. It may be easier for you experts to understand.

```
//** 'Complete Euler Squares' 5x5 of Pan-Diagonal Type **
//** File: 'CEuler5PD.c' Dictated by Kanji Setsuda **
//**  on Sep. 9, 2003; Revised on Feb.16, 2007; **
//**  Recompiled on Feb.14, 2015  **
//**  with MacOSX 10.10.2 and Xcode 6.1.1 **
//
#include <stdio.h>
//
short int cnt, cnt1;
short LSM, cnt2, cnt3;
short nm[26], uflg[26];
short anm[7][26];
short cl1h[6], rw1h[6], pd1h[6], pb1h[6];
short cl1l[6], rw1l[6], pd1l[6], pb1l[6];
short cl2h[6], rw2h[6], pd2h[6], pb2h[6];
short cl2l[6], rw2l[6], pd2l[6], pb2l[6];
short cl3h[6], rw3h[6], pd3h[6], pb3h[6];
short cl3l[6], rw3l[6], pd3l[6], pb3l[6];
short cl4h[6], rw4h[6], pd4h[6], pb4h[6];
short cl4l[6], rw4l[6], pd4l[6], pb4l[6];
short cl5h[6], rw5h[6], pd5h[6], pb5h[6];
short cl5l[6], rw5l[6], pd5l[6], pb5l[6];
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void stp09(void), stp10(void), stp11(void), stp12(void);
void stp13(void), stp14(void), stp15(void), stp16(void);
```

```
void stp17(void), stp18(void), stp19(void), stp20(void);
void stp21(void), stp22(void), stp23(void), stp24(void);
void stp25(void), ansprint(void), pr3ans(void);
void pr2ans(void), pr1ans(void);
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("*** 'Complete Euler Squares' of Order 5: Pan-Diagonal Type ***\n");
  printf("** Compact List of the 3600 Standard Solutions **\n");
  printf(" [Normalizing Inequality Condtions: n1<n25, n1<n5, n1<21 and n2>n6;]\n");
  printf("\n");
  for(n=0;n<26;n++){nm[n]=0; uflg[n]=0;}
  for(n=0;n<6;n++){
    cl1h[n]=0; rw1h[n]=0; pd1h[n]=0; pb1h[n]=0;
    cl1l[n]=0; rw1l[n]=0; pd1l[n]=0; pb1l[n]=0;
    cl2h[n]=0; rw2h[n]=0; pd2h[n]=0; pb2h[n]=0;
    cl2l[n]=0; rw2l[n]=0; pd2l[n]=0; pb2l[n]=0;
    cl3h[n]=0; rw3h[n]=0; pd3h[n]=0; pb3h[n]=0;
    cl3l[n]=0; rw3l[n]=0; pd3l[n]=0; pb3l[n]=0;
    cl4h[n]=0; rw4h[n]=0; pd4h[n]=0; pb4h[n]=0;
    cl4l[n]=0; rw4l[n]=0; pd4l[n]=0; pb4l[n]=0;
    cl5h[n]=0; rw5h[n]=0; pd5h[n]=0; pb5h[n]=0;
    cl5l[n]=0; rw5l[n]=0; pd5l[n]=0; pb5l[n]=0;}
  LSM=60; cnt=0; cnt3=0;
  stp01();      //* Begin the Calculations *
  if((0<cnt3)&&(cnt3<2)){pr1ans();}
  else if(cnt3>1){pr2ans();}
  printf("\n");
  printf("  [Total Solution Counts(n1=1/All) = %d/%d]  OK!\n",cnt1,cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda on\n");
  printf("    Feb.14, 2015 with MacOSX 10.10.2 and Xcode 6.1.1 **\n");
  printf("\n");
 return 0;
}
//
//* Begin the Calculations *
//* Set n1 *
void stp01(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl1h[a]==0)&&(rw1h[a]==0)&&(pd1h[a]==0)&&(pb1h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl1l[b]==0)&&(rw1l[b]==0)&&(pd1l[b]==0)&&(pb1l[b]==0)){
            nm[1]=d; uflg[d]=1; cnt2=0;
            cl1h[a]=1; rw1h[a]=1; pd1h[a]=1; pb1h[a]=1;
            cl1l[b]=1; rw1l[b]=1; pd1l[b]=1; pb1l[b]=1;
            stp02();
            cl1h[a]=0; rw1h[a]=0; pd1h[a]=0; pb1h[a]=0;
            cl1l[b]=0; rw1l[b]=0; pd1l[b]=0; pb1l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n25 & n1<n25 *
void stp02(){
 short a,b,d;
  for(a=4;a>=0;a--){
    if((cl5h[a]==0)&&(rw5h[a]==0)&&(pd1h[a]==0)&&(pb4h[a]==0)){
      for(b=4;b>=0;b--){d=a*5+b;
        if((nm[1]<d)&&(d<25)&&(uflg[d]==0)){
          if((cl5l[b]==0)&&(rw5l[b]==0)&&(pd1l[b]==0)&&(pb4l[b]==0)){
```

```
              nm[25]=d; uflg[d]=1; if(nm[1]==0){cnt2=0;}
              cl5h[a]=1; rw5h[a]=1; pd1h[a]=1; pb4h[a]=1;
              cl5l[b]=1; rw5l[b]=1; pd1l[b]=1; pb4l[b]=1;
              stp03();
              cl5h[a]=0; rw5h[a]=0; pd1h[a]=0; pb4h[a]=0;
              cl5l[b]=0; rw5l[b]=0; pd1l[b]=0; pb4l[b]=0;
              uflg[d]=0;}}
        }}
  }
}
//* Set n7 *
void stp03(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl2h[a]==0)&&(rw2h[a]==0)&&(pd1h[a]==0)&&(pb3h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl2l[b]==0)&&(rw2l[b]==0)&&(pd1l[b]==0)&&(pb3l[b]==0)){
            nm[7]=d; uflg[d]=1;
            cl2h[a]=1; rw2h[a]=1; pd1h[a]=1; pb3h[a]=1;
            cl2l[b]=1; rw2l[b]=1; pd1l[b]=1; pb3l[b]=1;
            stp04();
            cl2h[a]=0; rw2h[a]=0; pd1h[a]=0; pb3h[a]=0;
            cl2l[b]=0; rw2l[b]=0; pd1l[b]=0; pb3l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
//* Set n13 *
void stp04(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl3h[a]==0)&&(rw3h[a]==0)&&(pd1h[a]==0)&&(pb5h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl3l[b]==0)&&(rw3l[b]==0)&&(pd1l[b]==0)&&(pb5l[b]==0)){
            nm[13]=d; uflg[d]=1;
            cl3h[a]=1; rw3h[a]=1; pd1h[a]=1; pb5h[a]=1;
            cl3l[b]=1; rw3l[b]=1; pd1l[b]=1; pb5l[b]=1;
            stp05();
            cl3h[a]=0; rw3h[a]=0; pd1h[a]=0; pb5h[a]=0;
            cl3l[b]=0; rw3l[b]=0; pd1l[b]=0; pb5l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
//* Set n19=LSM-n1-n7-n13-n25 *
void stp05(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl4h[a]==0)&&(rw4h[a]==0)&&(pd1h[a]==0)&&(pb2h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[1]+nm[7]+nm[13]+d+nm[25]==LSM)&&(uflg[d]==0)){
          if((cl4l[b]==0)&&(rw4l[b]==0)&&(pd1l[b]==0)&&(pb2l[b]==0)){
            nm[19]=d; uflg[d]=1;
            cl4h[a]=1; rw4h[a]=1; pd1h[a]=1; pb2h[a]=1;
            cl4l[b]=1; rw4l[b]=1; pd1l[b]=1; pb2l[b]=1;
            stp06();
            cl4h[a]=0; rw4h[a]=0; pd1h[a]=0; pb2h[a]=0;
            cl4l[b]=0; rw4l[b]=0; pd1l[b]=0; pb2l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
//* Set n5 & n1<n5 *
```

```
void stp06(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl5h[a]==0)&&(rw1h[a]==0)&&(pd5h[a]==0)&&(pb5h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((d>nm[1])&&(uflg[d]==0)){
          if((cl5l[b]==0)&&(rw1l[b]==0)&&(pd5l[b]==0)&&(pb5l[b]==0)){
            nm[5]=d; uflg[d]=1;
            cl5h[a]=1; rw1h[a]=1; pd5h[a]=1; pb5h[a]=1;
            cl5l[b]=1; rw1l[b]=1; pd5l[b]=1; pb5l[b]=1;
            stp07();
            cl5h[a]=0; rw1h[a]=0; pd5h[a]=0; pb5h[a]=0;
            cl5l[b]=0; rw1l[b]=0; pd5l[b]=0; pb5l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n21 & n1<n21 *
void stp07(){
 short a,b,d;
  for(a=4;a>=0;a--){
    if((cl1h[a]==0)&&(rw5h[a]==0)&&(pd2h[a]==0)&&(pb5h[a]==0)){
      for(b=4;b>=0;b--){d=a*5+b;
        if((nm[1]<d)&&(uflg[d]==0)){
          if((cl1l[b]==0)&&(rw5l[b]==0)&&(pd2l[b]==0)&&(pb5l[b]==0)){
            nm[21]=d; uflg[d]=1;
            cl1h[a]=1; rw5h[a]=1; pd2h[a]=1; pb5h[a]=1;
            cl1l[b]=1; rw5l[b]=1; pd2l[b]=1; pb5l[b]=1;
            stp08();
            cl1h[a]=0; rw5h[a]=0; pd2h[a]=0; pb5h[a]=0;
            cl1l[b]=0; rw5l[b]=0; pd2l[b]=0; pb5l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n9 *
void stp08(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl4h[a]==0)&&(rw2h[a]==0)&&(pd3h[a]==0)&&(pb5h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl4l[b]==0)&&(rw2l[b]==0)&&(pd3l[b]==0)&&(pb5l[b]==0)){
            nm[9]=d; uflg[d]=1;
            cl4h[a]=1; rw2h[a]=1; pd3h[a]=1; pb5h[a]=1;
            cl4l[b]=1; rw2l[b]=1; pd3l[b]=1; pb5l[b]=1;
            stp09();
            cl4h[a]=0; rw2h[a]=0; pd3h[a]=0; pb5h[a]=0;
            cl4l[b]=0; rw2l[b]=0; pd3l[b]=0; pb5l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n17=LSM-n5-n9-n13-n21 *
void stp09(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl2h[a]==0)&&(rw4h[a]==0)&&(pd4h[a]==0)&&(pb5h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[5]+nm[9]+nm[13]+d+nm[21]==LSM)&&(uflg[d]==0)){
          if((cl2l[b]==0)&&(rw4l[b]==0)&&(pd4l[b]==0)&&(pb5l[b]==0)){
            nm[17]=d; uflg[d]=1;
            cl2h[a]=1; rw4h[a]=1; pd4h[a]=1; pb5h[a]=1;
            cl2l[b]=1; rw4l[b]=1; pd4l[b]=1; pb5l[b]=1;
            stp10();
```

```
              cl2h[a]=0; rw4h[a]=0; pd4h[a]=0; pb5h[a]=0;
              cl2l[b]=0; rw4l[b]=0; pd4l[b]=0; pb5l[b]=0;
              uflg[d]=0;}}
        }}
   }
}
//* Level #2: *
//* Set n2 *
void stp10(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl2h[a]==0)&&(rw1h[a]==0)&&(pd2h[a]==0)&&(pb2h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl2l[b]==0)&&(rw1l[b]==0)&&(pd2l[b]==0)&&(pb2l[b]==0)){
            nm[2]=d; uflg[d]=1;
            cl2h[a]=1; rw1h[a]=1; pd2h[a]=1; pb2h[a]=1;
            cl2l[b]=1; rw1l[b]=1; pd2l[b]=1; pb2l[b]=1;
            stp11();
            cl2h[a]=0; rw1h[a]=0; pd2h[a]=0; pb2h[a]=0;
            cl2l[b]=0; rw1l[b]=0; pd2l[b]=0; pb2l[b]=0;
            uflg[d]=0;}}
        }}
   }
}
//* Set n3 *
void stp11(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl3h[a]==0)&&(rw1h[a]==0)&&(pd3h[a]==0)&&(pb3h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl3l[b]==0)&&(rw1l[b]==0)&&(pd3l[b]==0)&&(pb3l[b]==0)){
            nm[3]=d; uflg[d]=1;
            cl3h[a]=1; rw1h[a]=1; pd3h[a]=1; pb3h[a]=1;
            cl3l[b]=1; rw1l[b]=1; pd3l[b]=1; pb3l[b]=1;
            stp12();
            cl3h[a]=0; rw1h[a]=0; pd3h[a]=0; pb3h[a]=0;
            cl3l[b]=0; rw1l[b]=0; pd3l[b]=0; pb3l[b]=0;
            uflg[d]=0;}}
        }}
   }
}
//* Set n4=LSM-n1-n2-n3-n5 *
void stp12(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl4h[a]==0)&&(rw1h[a]==0)&&(pd4h[a]==0)&&(pb4h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[1]+nm[2]+nm[3]+d+nm[5]==LSM)&&(uflg[d]==0)){
          if((cl4l[b]==0)&&(rw1l[b]==0)&&(pd4l[b]==0)&&(pb4l[b]==0)){
            nm[4]=d; uflg[d]=1;
            cl4h[a]=1; rw1h[a]=1; pd4h[a]=1; pb4h[a]=1;
            cl4l[b]=1; rw1l[b]=1; pd4l[b]=1; pb4l[b]=1;
            stp13();
            cl4h[a]=0; rw1h[a]=0; pd4h[a]=0; pb4h[a]=0;
            cl4l[b]=0; rw1l[b]=0; pd4l[b]=0; pb4l[b]=0;
            uflg[d]=0;}}
        }}
   }
}
//* Set n6 & n2>n6 *
void stp13(){
 short a,b,d;
  for(a=0;a<5;a++){
```

```
    if((cl1h[a]==0)&&(rw2h[a]==0)&&(pd5h[a]==0)&&(pb2h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((d<nm[2])&&(uflg[d]==0)){
          if((cl1l[b]==0)&&(rw2l[b]==0)&&(pd5l[b]==0)&&(pb2l[b]==0)){
            nm[6]=d; uflg[d]=1;
            cl1h[a]=1; rw2h[a]=1; pd5h[a]=1; pb2h[a]=1;
            cl1l[b]=1; rw2l[b]=1; pd5l[b]=1; pb2l[b]=1;
            stp14();
            cl1h[a]=0; rw2h[a]=0; pd5h[a]=0; pb2h[a]=0;
            cl1l[b]=0; rw2l[b]=0; pd5l[b]=0; pb2l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n11 *
void stp14(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl1h[a]==0)&&(rw3h[a]==0)&&(pd4h[a]==0)&&(pb3h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl1l[b]==0)&&(rw3l[b]==0)&&(pd4l[b]==0)&&(pb3l[b]==0)){
            nm[11]=d; uflg[d]=1;
            cl1h[a]=1; rw3h[a]=1; pd4h[a]=1; pb3h[a]=1;
            cl1l[b]=1; rw3l[b]=1; pd4l[b]=1; pb3l[b]=1;
            stp15();
            cl1h[a]=0; rw3h[a]=0; pd4h[a]=0; pb3h[a]=0;
            cl1l[b]=0; rw3l[b]=0; pd4l[b]=0; pb3l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n16=LSM-n1-n6-n11-n21 *
void stp15(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl1h[a]==0)&&(rw4h[a]==0)&&(pd3h[a]==0)&&(pb4h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[1]+nm[6]+nm[11]+d+nm[21]==LSM)&&(uflg[d]==0)){
          if((cl1l[b]==0)&&(rw4l[b]==0)&&(pd3l[b]==0)&&(pb4l[b]==0)){
            nm[16]=d; uflg[d]=1;
            cl1h[a]=1; rw4h[a]=1; pd3h[a]=1; pb4h[a]=1;
            cl1l[b]=1; rw4l[b]=1; pd3l[b]=1; pb4l[b]=1;
            stp16();
            cl1h[a]=0; rw4h[a]=0; pd3h[a]=0; pb4h[a]=0;
            cl1l[b]=0; rw4l[b]=0; pd3l[b]=0; pb4l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n8 *
void stp16(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl3h[a]==0)&&(rw2h[a]==0)&&(pd2h[a]==0)&&(pb4h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if(uflg[d]==0){
          if((cl3l[b]==0)&&(rw2l[b]==0)&&(pd2l[b]==0)&&(pb4l[b]==0)){
            nm[8]=d; uflg[d]=1;
            cl3h[a]=1; rw2h[a]=1; pd2h[a]=1; pb4h[a]=1;
            cl3l[b]=1; rw2l[b]=1; pd2l[b]=1; pb4l[b]=1;
            stp17();
            cl3h[a]=0; rw2h[a]=0; pd2h[a]=0; pb4h[a]=0;
            cl3l[b]=0; rw2l[b]=0; pd2l[b]=0; pb4l[b]=0;
            uflg[d]=0;}}
```

```
      }}
  }
}
//* Set n12=LSM-n4-n8-n16-n25 *
void stp17(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl2h[a]==0)&&(rw3h[a]==0)&&(pd5h[a]==0)&&(pb4h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[4]+nm[8]+d+nm[16]+nm[25]==LSM)&&(uflg[d]==0)){
          if((cl2l[b]==0)&&(rw3l[b]==0)&&(pd5l[b]==0)&&(pb4l[b]==0)){
            nm[12]=d; uflg[d]=1;
            cl2h[a]=1; rw3h[a]=1; pd5h[a]=1; pb4h[a]=1;
            cl2l[b]=1; rw3l[b]=1; pd5l[b]=1; pb4l[b]=1;
            stp18();
            cl2h[a]=0; rw3h[a]=0; pd5h[a]=0; pb4h[a]=0;
            cl2l[b]=0; rw3l[b]=0; pd5l[b]=0; pb4l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n10=LSM-n6-n7-n8-n9 *
void stp18(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl5h[a]==0)&&(rw2h[a]==0)&&(pd4h[a]==0)&&(pb1h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[6]+nm[7]+nm[8]+nm[9]+d==LSM)&&(uflg[d]==0)){
          if((cl5l[b]==0)&&(rw2l[b]==0)&&(pd4l[b]==0)&&(pb1l[b]==0)){
            nm[10]=d; uflg[d]=1;
            cl5h[a]=1; rw2h[a]=1; pd4h[a]=1; pb1h[a]=1;
            cl5l[b]=1; rw2l[b]=1; pd4l[b]=1; pb1l[b]=1;
            stp19();
            cl5h[a]=0; rw2h[a]=0; pd4h[a]=0; pb1h[a]=0;
            cl5l[b]=0; rw2l[b]=0; pd4l[b]=0; pb1l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n22=LSM-n2-n7-n12-n17 *
void stp19(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl2h[a]==0)&&(rw5h[a]==0)&&(pd3h[a]==0)&&(pb1h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[2]+nm[7]+nm[12]+nm[17]+d==LSM)&&(uflg[d]==0)){
          if((cl2l[b]==0)&&(rw5l[b]==0)&&(pd3l[b]==0)&&(pb1l[b]==0)){
            nm[22]=d; uflg[d]=1;
            cl2h[a]=1; rw5h[a]=1; pd3h[a]=1; pb1h[a]=1;
            cl2l[b]=1; rw5l[b]=1; pd3l[b]=1; pb1l[b]=1;
            stp20();
            cl2h[a]=0; rw5h[a]=0; pd3h[a]=0; pb1h[a]=0;
            cl2l[b]=0; rw5l[b]=0; pd3l[b]=0; pb1l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Level #3: *
//* Set n23=LSM-n4-n10-n11-n17 *
void stp20(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl3h[a]==0)&&(rw5h[a]==0)&&(pd4h[a]==0)&&(pb2h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[4]+nm[10]+nm[11]+nm[17]+d==LSM)&&(uflg[d]==0)){
```

```
            if((cl3l[b]==0)&&(rw5l[b]==0)&&(pd4l[b]==0)&&(pb2l[b]==0)){
              nm[23]=d; uflg[d]=1;
              cl3h[a]=1; rw5h[a]=1; pd4h[a]=1; pb2h[a]=1;
              cl3l[b]=1; rw5l[b]=1; pd4l[b]=1; pb2l[b]=1;
              stp21();
              cl3h[a]=0; rw5h[a]=0; pd4h[a]=0; pb2h[a]=0;
              cl3l[b]=0; rw5l[b]=0; pd4l[b]=0; pb2l[b]=0;
              uflg[d]=0;}}
        }}
  }
}
//* Set n18=LSM-n3-n8-n13-n23 *
void stp21(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl3h[a]==0)&&(rw4h[a]==0)&&(pd5h[a]==0)&&(pb1h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[3]+nm[8]+nm[13]+d+nm[23]==LSM)&&(uflg[d]==0)){
          if((cl3l[b]==0)&&(rw4l[b]==0)&&(pd5l[b]==0)&&(pb1l[b]==0)){
            nm[18]=d; uflg[d]=1;
            cl3h[a]=1; rw4h[a]=1; pd5h[a]=1; pb1h[a]=1;
            cl3l[b]=1; rw4l[b]=1; pd5l[b]=1; pb1l[b]=1;
            stp22();
            cl3h[a]=0; rw4h[a]=0; pd5h[a]=0; pb1h[a]=0;
            cl3l[b]=0; rw4l[b]=0; pd5l[b]=0; pb1l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
//* Set n14=LSM-n1-n10-n18-n22 *
void stp22(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl4h[a]==0)&&(rw3h[a]==0)&&(pd2h[a]==0)&&(pb1h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[1]+nm[10]+d+nm[18]+nm[22]==LSM)&&(uflg[d]==0)){
          if((cl4l[b]==0)&&(rw3l[b]==0)&&(pd2l[b]==0)&&(pb1l[b]==0)){
            nm[14]=d; uflg[d]=1;
            cl4h[a]=1; rw3h[a]=1; pd2h[a]=1; pb1h[a]=1;
            cl4l[b]=1; rw3l[b]=1; pd2l[b]=1; pb1l[b]=1;
            stp23();
            cl4h[a]=0; rw3h[a]=0; pd2h[a]=0; pb1h[a]=0;
            cl4l[b]=0; rw3l[b]=0; pd2l[b]=0; pb1l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
//* Set n15=LSM-n3-n9-n16-n22 *
void stp23(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl5h[a]==0)&&(rw3h[a]==0)&&(pd3h[a]==0)&&(pb2h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[3]+nm[9]+d+nm[16]+nm[22]==LSM)&&(uflg[d]==0)){
          if((cl5l[b]==0)&&(rw3l[b]==0)&&(pd3l[b]==0)&&(pb2l[b]==0)){
            nm[15]=d; uflg[d]=1;
            cl5h[a]=1; rw3h[a]=1; pd3h[a]=1; pb2h[a]=1;
            cl5l[b]=1; rw3l[b]=1; pd3l[b]=1; pb2l[b]=1;
            stp24();
            cl5h[a]=0; rw3h[a]=0; pd3h[a]=0; pb2h[a]=0;
            cl5l[b]=0; rw3l[b]=0; pd3l[b]=0; pb2l[b]=0;
            uflg[d]=0;}}
        }}
  }
}
```

```
//* Set n20=LSM-n2-n8-n14-n21 *
void stp24(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl5h[a]==0)&&(rw4h[a]==0)&&(pd2h[a]==0)&&(pb3h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[2]+nm[8]+nm[14]+d+nm[21]==LSM)&&(uflg[d]==0)){
          if((cl5l[b]==0)&&(rw4l[b]==0)&&(pd2l[b]==0)&&(pb3l[b]==0)){
            nm[20]=d; uflg[d]=1;
            cl5h[a]=1; rw4h[a]=1; pd2h[a]=1; pb3h[a]=1;
            cl5l[b]=1; rw4l[b]=1; pd2l[b]=1; pb3l[b]=1;
            stp25();
            cl5h[a]=0; rw4h[a]=0; pd2h[a]=0; pb3h[a]=0;
            cl5l[b]=0; rw4l[b]=0; pd2l[b]=0; pb3l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//* Set n24=LSM-n3-n7-n11-n20 *
void stp25(){
 short a,b,d;
  for(a=0;a<5;a++){
    if((cl4h[a]==0)&&(rw5h[a]==0)&&(pd5h[a]==0)&&(pb3h[a]==0)){
      for(b=0;b<5;b++){d=a*5+b;
        if((nm[3]+nm[7]+nm[11]+nm[20]+d==LSM)&&(uflg[d]==0)){
          if((cl4l[b]==0)&&(rw5l[b]==0)&&(pd5l[b]==0)&&(pb3l[b]==0)){
            nm[24]=d; uflg[d]=1;
            cl4h[a]=1; rw5h[a]=1; pd5h[a]=1; pb3h[a]=1;
            cl4l[b]=1; rw5l[b]=1; pd5l[b]=1; pb3l[b]=1;
            ansprint();
            cl4h[a]=0; rw5h[a]=0; pd5h[a]=0; pb3h[a]=0;
            cl4l[b]=0; rw5l[b]=0; pd5l[b]=0; pb3l[b]=0;
            uflg[d]=0;}}
      }}
  }
}
//
//* Print the Answers *
void ansprint(){
 short n;
  cnt++; if(nm[1]==1){cnt1++;}
  cnt2++;
  if(cnt2==1){
    anm[cnt3*2][0]=cnt;
    for(n=1;n<26;n++){anm[cnt3*2][n]=nm[n]+1; anm[cnt3*2+1][n]=nm[n];}
    cnt3++;
    if(cnt3==3){pr3ans(); cnt3=0;
      anm[2][0]=0; anm[4][0]=0;
      for(n=1;n<26;n++){anm[2][n]=0; anm[3][n]=0;}
    }
  }
}
//
//* Print 3 Answers *
void pr3ans(){
 short l,l5,n;
  printf(" /D5i%22d#   /D5i%22d#   /D5i%22d#\n",
          anm[0][0],anm[2][0],anm[4][0]);
  for(l=0;l<5;l++){l5=l*5;
    printf(" ");
    for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])/5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])%5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%3d",anm[0][l5+n]);}
```

```
      printf("   ");
      for(n=1;n<6;n++){printf("%d",(anm[3][l5+n])/5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[3][l5+n])%5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%3d",anm[2][l5+n]);}
      printf("   ");
      for(n=1;n<6;n++){printf("%d",(anm[5][l5+n])/5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[5][l5+n])%5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%3d",anm[4][l5+n]);}
      printf("\n");
    }
}
//
//* Print 2 Answers *
void pr2ans(){
 short l,l5,n;
   printf(" /D5i%22d#   /D5i%22d#\n",anm[0][0],anm[2][0]);
   for(l=0;l<5;l++){l5=l*5;
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])/5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])%5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%3d",anm[0][l5+n]);}
      printf("   ");
      for(n=1;n<6;n++){printf("%d",(anm[3][l5+n])/5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[3][l5+n])%5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%3d",anm[2][l5+n]);}
      printf("\n");
   }
}
//
//* Print 1 Answer *
void pr1ans(){
 short l,l5,n;
   printf(" /D5i%22d#\n",anm[0][0]);
   for(l=0;l<5;l++){l5=l*5;
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])/5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%d",(anm[1][l5+n])%5);}
      printf(" ");
      for(n=1;n<6;n++){printf("%3d",anm[0][l5+n]);}
      printf("\n");
   }
}
//
```

#5. Result List of My Recent Computation

```
** 'Complete Euler Squares' of Order 5: Pan-Diagonal Type **
   ** Compact List of the 3600 Standard Solutions **
 [Normalizing Inequality Condtions: n1<n25, n1<n5, n1<21 and n2>n6;]

 /D5i                    1#   /D5i                   37#   /D5i                   73#
 04321 02413   1 23 20 12  9   04321 02314   1 23 19 12 10   04321 03214   1 24 18 12 10
 21043 41302  15  7  4 21 18   21043 31402  14  7  5 21 18   21043 21403  13  7  5 21 19
 43210 30241  24 16 13 10  2   43210 40231  25 16 13  9  2   43210 40321  25 16 14  8  2
 10432 24130   8  5 22 19 11   10432 23140   8  4 22 20 11   10432 32140   9  3 22 20 11
 32104 13024  17 14  6  3 25   32104 14023  17 15  6  3 24   32104 14032  17 15  6  4 23
```

```
/D5i                    109#
04321 03124   1 24 17 13 10
21043 12403  12  8  5 21 19
43210 40312  25 16 14  7  3
10432 31240   9  2 23 20 11
32104 24031  18 15  6  4 22
/D5i                    145#
03421 02413   1 18 25 12  9
21034 41302  15  7  4 16 23
34210 30241  19 21 13 10  2
10342 24130   8  5 17 24 11
42103 13024  22 14  6  3 20
/D5i                    181#
03421 02314   1 18 24 12 10
21034 31402  14  7  5 16 23
34210 40231  20 21 13  9  2
10342 23140   8  4 17 25 11
42103 14023  22 15  6  3 19
/D5i                    217#
03421 03214   1 19 23 12 10
21034 21403  13  7  5 16 24
34210 40321  20 21 14  8  2
10342 32140   9  3 17 25 11
42103 14032  22 15  6  4 18
/D5i                    253#
03421 03124   1 19 22 13 10
21034 12403  12  8  5 16 24
34210 40312  20 21 14  7  3
10342 31240   9  2 18 25 11
42103 24031  23 15  6  4 17
/D5i                    289#
03214 04321   1 20 14  8 22
21403 21043  13  7 21  5 19
40321 43210  25  4 18 12  6
32140 10432  17 11 10 24  3
14032 32104   9 23  2 16 15
/D5i                    325#
03214 03421   1 19 15  8 22
21403 21034  13  7 21  4 20
40321 34210  24  5 18 12  6
32140 10342  17 11  9 25  3
14032 42103  10 23  2 16 14
/D5i                    361#
03214 02431   1 18 15  9 22
21403 31024  14  7 21  3 20
40321 24310  23  5 19 12  6
32140 10243  17 11  8 25  4
14032 43102  10 24  2 16 13
/D5i                    397#
03214 01432   1 17 15  9 23
21403 32014  14  8 21  2 20
40321 14320  22  5 19 13  6
32140 20143  18 11  7 25  4
14032 43201  10 24  3 16 12
/D5i                    433#
03124 04321   1 20  9 13 22
12403 21043   8 12 21  5 19
40312 43210  25  4 18  7 11
31240 10432  17  6 15 24  3
24031 32104  14 23  2 16 10
/D5i                    469#
03124 03421   1 19 10 13 22
12403 21034   8 12 21  4 20
40312 34210  24  5 18  7 11
31240 10342  17  6 14 25  3
24031 42103  15 23  2 16  9
/D5i                    505#
03124 02431   1 18 10 14 22
12403 31024   9 12 21  3 20
40312 24310  23  5 19  7 11
31240 10243  17  6 13 25  4
24031 43102  15 24  2 16  8
/D5i                    541#
03124 01432   1 17 10 14 23
12403 32014   9 13 21  2 20
40312 14320  22  5 19  8 11
31240 20143  18  6 12 25  4
24031 43201  15 24  3 16  7
/D5i                    577#
04321 12403   2 23 20 11  9
21043 40312  15  6  4 22 18
43210 31240  24 17 13 10  1
10432 24031   8  5 21 19 12
32104 03124  16 14  7  3 25
/D5i                   1153#
04321 21403   3 22 20 11  9
21043 40321  15  6  4 23 17
43210 32140  24 18 12 10  1
10432 14032   7  5 21 19 13
32104 03214  16 14  8  2 25
/D5i                   1729#
04321 31402   4 22 20 11  8
21043 40231  15  6  3 24 17
43210 23140  23 19 12 10  1
10432 14023   7  5 21 18 14
32104 02314  16 13  9  2 25
/D5i                   2305#
04321 41302   5 22 19 11  8
21043 30241  14  6  3 25 17
43210 24130  23 20 12  9  1
10432 13024   7  4 21 18 15
32104 02413  16 13 10  2 24
/D5i                   2881#
14302 02413   6 23 20  2 14
02143 41302   5 12  9 21 18
43021 30241  24 16  3 15  7
21430 24130  13 10 22 19  1
30214 13024  17  4 11  8 25
/D5i                   3025#
14302 12403   7 23 20  1 14
02143 40312   5 11  9 22 18
43021 31240  24 17  3 15  6
21430 24031  13 10 21 19  2
30214 03124  16  4 12  8 25
/D5i                   3169#
14302 21403   8 22 20  1 14
02143 40321   5 11  9 23 17
43021 32140  24 18  2 15  6
21430 14032  12 10 21 19  3
30214 03214  16  4 13  7 25
/D5i                   3313#
14302 31402   9 22 20  1 13
02143 40231   5 11  8 24 17
43021 23140  23 19  2 15  6
21430 14023  12 10 21 18  4
30214 02314  16  3 14  7 25
/D5i                   3457#
14302 41302  10 22 19  1 13
02143 30241   4 11  8 25 17
43021 24130  23 20  2 14  6
21430 13024  12  9 21 18  5
30214 02413  16  3 15  7 24

  [Total Solution Counts(n1=1/All) = 576/3600]  OK!

** Calculated and Listed by Kanji Setsuda on
    Feb.14, 2015 with MacOSX 10.10.2 and Xcode 6.1.1 **
```

We already know about these 3600 Pan-diagonal Magic Squares of Order 5. They were once calculated and announced by our Prof. Mutsumi Suzuki in his famous web page. We also know that all of them are 'Complete Euler Squares.'

What we have now got are just the same with those 'Suzuki Squares.'

Therefore, we do not only know our program could work all right, but also know it could verify well that all of 'Suzuki Squares 5x5' are really 'Complete Euler Squares.'

How glad I am to know that!

I was deeply impressed by the unexpected high speed of our new program, though I am afraid it might look too complicated for you to try to do with it by yourself.

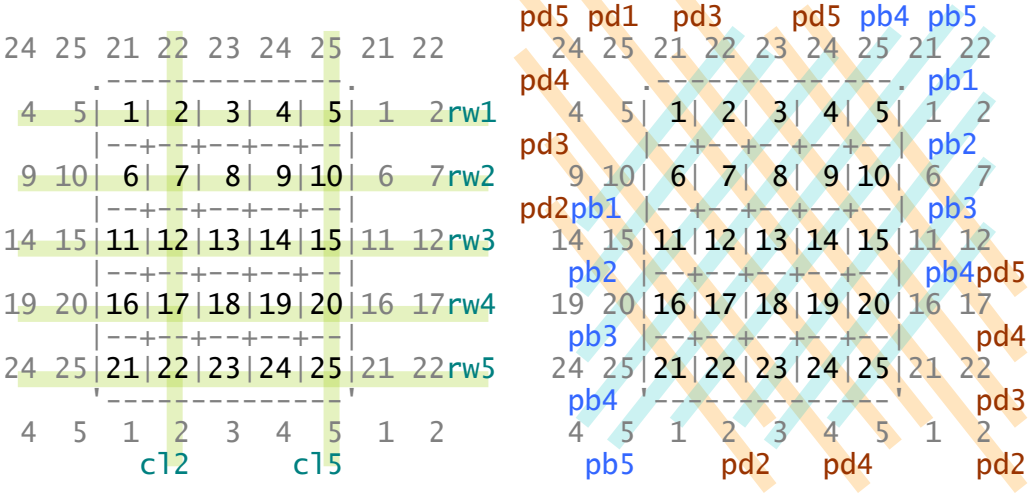
#6. How about The 16 Simultaneous Magic Squares: Both SC and PD Type?

Can we make another 'Suzuki Squares' by our new method of programming? I mean the 16 simultaneous magic squares 5x5: both self-complementary and pan-diagonal.

I must report I could make them, by determining each complementary pair in the same procedure. When you determine `n1`, you can calculate `n25` at the same time by such a simple subtraction as `n25=CC-n1`.

You can also determine all the other complementary pairs `(n2, n24)`, `(n3, n23)`, `(n4, n22)`, `...`, `(n11, n15)` and `(n12, n14)` at the same time in the same way.

```
**  Basic Forms for Simultaneous Magic Squares of Order 5  **
```




```
* Basic Conditions: C=60; *
  n1+n2+n3+n4+n5=C       ... rw1 | n1+n6+n11+n16+n21=C  ... cl1
  n6+n7+n8+n9+n10=C      ... rw2 | n2+n7+n12+n17+n22=C  ... cl2
  n11+n12+n13+n14+n15=C  ... rw3 | n3+n8+n13+n18+n23=C  ... cl3
  n16+n17+n18+n19+n20=C  ... rw4 | n4+n9+n14+n19+n24=C  ... cl4
  n21+n22+n23+n24+n25=C  ... rw5 | n5+n10+n15+n20+n25=C ... cl5
* Pandiagonal Conditions: C=60; *
  n1+n7+n13+n19+n25=C    ... pd1 | n1+n10+n14+n18+n22=C ... pb1
  n2+n8+n14+n20+n21=C    ... pd2 | n2+n6+n15+n19+n23=C  ... pb2
  n3+n9+n15+n16+n22=C    ... pd3 | n3+n7+n11+n20+n24=C  ... pb3
  n4+n10+n11+n17+n23=C   ... pd4 | n4+n8+n12+n16+n25=C  ... pb4
  n5+n6+n12+n18+n24=C    ... pd5 | n5+n9+n13+n17+n21=C  ... pb5
* Self-Complementary Conditions: CC=24; *
  n1+n25=n2+n24=n3+n23=n4+n22=n5+n21=n6+n20=n7+n19
  =n8+n18=n9+n17=n10+n16=n11+n15=n12+n14=n13+n13=CC  ...(sc)
* List-forming Inequality Conditions: *
  n1<n25; n1<n5; n1<n21; and n2>n6;
```

#7. Program List

```
//*** Compose 'Complete Euler Squares' of Order 5: ***
//** Simultaneous Type: Both Self-Complementary and Pan-Diagonal **
//** File: 'CEuler5Sml.c' Dictated by Kanji Setsuda in 2003; **
//**  Revised on Feb.16, 2007; Recompiled on Feb.14, 2015  **
//**  working with MacOSX 10.10.2 and Xcode 6.1.1; **
//
#include <stdio.h>
//
short int cnt, cnt1, cnt3;
```

```
short LSM, CC;
short nm[26], uflg[26];
short anm[5][26];
short cl1h[6], rw1h[6], pd1h[6], pb1h[6];
short cl1l[6], rw1l[6], pd1l[6], pb1l[6];
short cl2h[6], rw2h[6], pd2h[6], pb2h[6];
short cl2l[6], rw2l[6], pd2l[6], pb2l[6];
short cl3h[6], rw3h[6], pd3h[6], pb3h[6];
short cl3l[6], rw3l[6], pd3l[6], pb3l[6];
short cl4h[6], rw4h[6], pd4h[6], pb4h[6];
short cl4l[6], rw4l[6], pd4l[6], pb4l[6];
short cl5h[6], rw5h[6], pd5h[6], pb5h[6];
short cl5l[6], rw5l[6], pd5l[6], pb5l[6];
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void stp09(void), stp10(void), stp11(void), stp12(void);
void ansprint(void), prans(void);
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("*** 'Complete Euler Squares' of Order 5:\n");
  printf("  Simultaneous Type: Both Self-Complementary and Pan-Diagonal **\n");
  printf("** List of the 16 Standard Solutions **\n");
  printf(" [Normalizing Inequality Condtions: n1<n25, n1<n5, n1<21 and n2>n6;]\n");
  printf("\n");
  for(n=0;n<26;n++){nm[n]=0; uflg[n]=0;}
  for(n=0;n<6;n++){
    cl1h[n]=0; rw1h[n]=0; pd1h[n]=0; pb1h[n]=0;
    cl1l[n]=0; rw1l[n]=0; pd1l[n]=0; pb1l[n]=0;
    cl2h[n]=0; rw2h[n]=0; pd2h[n]=0; pb2h[n]=0;
    cl2l[n]=0; rw2l[n]=0; pd2l[n]=0; pb2l[n]=0;
    cl3h[n]=0; rw3h[n]=0; pd3h[n]=0; pb3h[n]=0;
    cl3l[n]=0; rw3l[n]=0; pd3l[n]=0; pb3l[n]=0;
    cl4h[n]=0; rw4h[n]=0; pd4h[n]=0; pb4h[n]=0;
    cl4l[n]=0; rw4l[n]=0; pd4l[n]=0; pb4l[n]=0;
    cl5h[n]=0; rw5h[n]=0; pd5h[n]=0; pb5h[n]=0;
    cl5l[n]=0; rw5l[n]=0; pd5l[n]=0; pb5l[n]=0;}
  LSM=60; CC=24; cnt=0; cnt3=0;
  nm[13]=12; uflg[12]=1;
  cl3h[2]=1; rw3h[2]=1; pd1h[2]=1; pb5h[2]=1;
  cl3l[2]=1; rw3l[2]=1; pd1l[2]=1; pb5l[2]=1;
  stp01();      //* Begin The Calculations *
  if(cnt3>0){prans();}
  printf("\n");
  printf("  [Total Solution Counts(n1=1/All) = %d/%d]  OK!\n",cnt1,cnt);
  printf("\n");
  printf("** Calculated and Listed by Kanji Setsuda on\n");
  printf("    Feb.14, 2015 with MacOSX 10.10.2 and Xcode 6.1.1 **\n");
  printf("\n");
 return 0;
}
//
//* Begin The Calculations *
//* Set n1 & n25 & n1<n25 *
void stp01(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl1h[a]==0)&&(rw1h[a]==0)&&(pd1h[a]==0)&&(pb1h[a]==0)){
     if((cl5h[na]==0)&&(rw5h[na]==0)&&(pd1h[na]==0)&&(pb4h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((uflg[d]==0)&&(uflg[nd]==0)&&(d<nd)){
           if((cl1l[b]==0)&&(rw1l[b]==0)&&(pd1l[b]==0)&&(pb1l[b]==0)){
```

```
              if((cl5l[nb]==0)&&(rw5l[nb]==0)&&(pd1l[nb]==0)&&(pb4l[nb]==0)){
                nm[1]=d; nm[25]=nd; uflg[d]=1; uflg[nd]=1;
                cl1h[a]=1; rw1h[a]=1; pd1h[a]=1; pb1h[a]=1;
                cl1l[b]=1; rw1l[b]=1; pd1l[b]=1; pb1l[b]=1;
                cl5h[na]=1; rw5h[na]=1; pd1h[na]=1; pb4h[na]=1;
                cl5l[nb]=1; rw5l[nb]=1; pd1l[nb]=1; pb4l[nb]=1;
                stp02();
                cl5h[na]=0; rw5h[na]=0; pd1h[na]=0; pb4h[na]=0;
                cl5l[nb]=0; rw5l[nb]=0; pd1l[nb]=0; pb4l[nb]=0;
                cl1h[a]=0; rw1h[a]=0; pd1h[a]=0; pb1h[a]=0;
                cl1l[b]=0; rw1l[b]=0; pd1l[b]=0; pb1l[b]=0;
                uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n7 & n19 & n1+n7+n13+n19+n25=LSM? *
void stp02(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl2h[a]==0)&&(rw2h[a]==0)&&(pd1h[a]==0)&&(pb3h[a]==0)){
     if((cl4h[na]==0)&&(rw4h[na]==0)&&(pd1h[na]==0)&&(pb2h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[1]+d+nm[13]+nd+nm[25]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl2l[b]==0)&&(rw2l[b]==0)&&(pd1l[b]==0)&&(pb3l[b]==0)){
             if((cl4l[nb]==0)&&(rw4l[nb]==0)&&(pd1l[nb]==0)&&(pb2l[nb]==0)){
               nm[7]=d; nm[19]=nd; uflg[d]=1; uflg[nd]=1;
               cl2h[a]=1; rw2h[a]=1; pd1h[a]=1; pb3h[a]=1;
               cl2l[b]=1; rw2l[b]=1; pd1l[b]=1; pb3l[b]=1;
               cl4h[na]=1; rw4h[na]=1; pd1h[na]=1; pb2h[na]=1;
               cl4l[nb]=1; rw4l[nb]=1; pd1l[nb]=1; pb2l[nb]=1;
               stp03();
               cl4h[na]=0; rw4h[na]=0; pd1h[na]=0; pb2h[na]=0;
               cl4l[nb]=0; rw4l[nb]=0; pd1l[nb]=0; pb2l[nb]=0;
               cl2h[a]=0; rw2h[a]=0; pd1h[a]=0; pb3h[a]=0;
               cl2l[b]=0; rw2l[b]=0; pd1l[b]=0; pb3l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n5 & & n21 & n1<n5 & n1<n21 *
void stp03(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl5h[a]==0)&&(rw1h[a]==0)&&(pd5h[a]==0)&&(pb5h[a]==0)){
     if((cl1h[na]==0)&&(rw5h[na]==0)&&(pd2h[na]==0)&&(pb5h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[1]<d)&&(nm[1]<nd)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl5l[b]==0)&&(rw1l[b]==0)&&(pd5l[b]==0)&&(pb5l[b]==0)){
             if((cl1l[nb]==0)&&(rw5l[nb]==0)&&(pd2l[nb]==0)&&(pb5l[nb]==0)){
               nm[5]=d; nm[21]=nd; uflg[d]=1; uflg[nd]=1;
               cl5h[a]=1; rw1h[a]=1; pd5h[a]=1; pb5h[a]=1;
               cl5l[b]=1; rw1l[b]=1; pd5l[b]=1; pb5l[b]=1;
               cl1h[na]=1; rw5h[na]=1; pd2h[na]=1; pb5h[na]=1;
               cl1l[nb]=1; rw5l[nb]=1; pd2l[nb]=1; pb5l[nb]=1;
               stp04();
               cl1h[na]=0; rw5h[na]=0; pd2h[na]=0; pb5h[na]=0;
               cl1l[nb]=0; rw5l[nb]=0; pd2l[nb]=0; pb5l[nb]=0;
               cl5h[a]=0; rw1h[a]=0; pd5h[a]=0; pb5h[a]=0;
               cl5l[b]=0; rw1l[b]=0; pd5l[b]=0; pb5l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n9 & n17 & n5+n9+n13+n17+n21=LSM? *
void stp04(){
```

```
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl4h[a]==0)&&(rw2h[a]==0)&&(pd3h[a]==0)&&(pb5h[a]==0)){
     if((cl2h[na]==0)&&(rw4h[na]==0)&&(pd4h[na]==0)&&(pb5h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[5]+d+nm[13]+nd+nm[21]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl4l[b]==0)&&(rw2l[b]==0)&&(pd3l[b]==0)&&(pb5l[b]==0)){
             if((cl2l[nb]==0)&&(rw4l[nb]==0)&&(pd4l[nb]==0)&&(pb5l[nb]==0)){
               nm[9]=d; nm[17]=nd; uflg[d]=1; uflg[nd]=1;
               cl4h[a]=1; rw2h[a]=1; pd3h[a]=1; pb5h[a]=1;
               cl4l[b]=1; rw2l[b]=1; pd3l[b]=1; pb5l[b]=1;
               cl2h[na]=1; rw4h[na]=1; pd4h[na]=1; pb5h[na]=1;
               cl2l[nb]=1; rw4l[nb]=1; pd4l[nb]=1; pb5l[nb]=1;
               stp05();
               cl2h[na]=0; rw4h[na]=0; pd4h[na]=0; pb5h[na]=0;
               cl2l[nb]=0; rw4l[nb]=0; pd4l[nb]=0; pb5l[nb]=0;
               cl4h[a]=0; rw2h[a]=0; pd3h[a]=0; pb5h[a]=0;
               cl4l[b]=0; rw2l[b]=0; pd3l[b]=0; pb5l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
       }}}
  }
}
//* Level #2: *
//* Set n2 & n24 *
void stp05(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl2h[a]==0)&&(rw1h[a]==0)&&(pd2h[a]==0)&&(pb2h[a]==0)){
     if((cl4h[na]==0)&&(rw5h[na]==0)&&(pd5h[na]==0)&&(pb3h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((uflg[d]==0)&&(uflg[nd]==0)){
           if((cl2l[b]==0)&&(rw1l[b]==0)&&(pd2l[b]==0)&&(pb2l[b]==0)){
             if((cl4l[nb]==0)&&(rw5l[nb]==0)&&(pd5l[nb]==0)&&(pb3l[nb]==0)){
               nm[2]=d; nm[24]=nd; uflg[d]=1; uflg[nd]=1;
               cl2h[a]=1; rw1h[a]=1; pd2h[a]=1; pb2h[a]=1;
               cl2l[b]=1; rw1l[b]=1; pd2l[b]=1; pb2l[b]=1;
               cl4h[na]=1; rw5h[na]=1; pd5h[na]=1; pb3h[na]=1;
               cl4l[nb]=1; rw5l[nb]=1; pd5l[nb]=1; pb3l[nb]=1;
               stp06();
               cl4h[na]=0; rw5h[na]=0; pd5h[na]=0; pb3h[na]=0;
               cl4l[nb]=0; rw5l[nb]=0; pd5l[nb]=0; pb3l[nb]=0;
               cl2h[a]=0; rw1h[a]=0; pd2h[a]=0; pb2h[a]=0;
               cl2l[b]=0; rw1l[b]=0; pd2l[b]=0; pb2l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
       }}}
  }
}
//* Set n3 & n23 *
void stp06(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl3h[a]==0)&&(rw1h[a]==0)&&(pd3h[a]==0)&&(pb3h[a]==0)){
     if((cl3h[na]==0)&&(rw5h[na]==0)&&(pd4h[na]==0)&&(pb2h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((uflg[d]==0)&&(uflg[nd]==0)){
           if((cl3l[b]==0)&&(rw1l[b]==0)&&(pd3l[b]==0)&&(pb3l[b]==0)){
             if((cl3l[nb]==0)&&(rw5l[nb]==0)&&(pd4l[nb]==0)&&(pb2l[nb]==0)){
               nm[3]=d; nm[23]=nd; uflg[d]=1; uflg[nd]=1;
               cl3h[a]=1; rw1h[a]=1; pd3h[a]=1; pb3h[a]=1;
               cl3l[b]=1; rw1l[b]=1; pd3l[b]=1; pb3l[b]=1;
               cl3h[na]=1; rw5h[na]=1; pd4h[na]=1; pb2h[na]=1;
               cl3l[nb]=1; rw5l[nb]=1; pd4l[nb]=1; pb2l[nb]=1;
               stp07();
               cl3h[na]=0; rw5h[na]=0; pd4h[na]=0; pb2h[na]=0;
               cl3l[nb]=0; rw5l[nb]=0; pd4l[nb]=0; pb2l[nb]=0;
```

```
                cl3h[a]=0; rw1h[a]=0; pd3h[a]=0; pb3h[a]=0;
                cl3l[b]=0; rw1l[b]=0; pd3l[b]=0; pb3l[b]=0;
                uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n4=LSM-n1-n2-n3-n5 & n22 *
void stp07(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl4h[a]==0)&&(rw1h[a]==0)&&(pd4h[a]==0)&&(pb4h[a]==0)){
     if((cl2h[na]==0)&&(rw5h[na]==0)&&(pd3h[na]==0)&&(pb1h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[1]+nm[2]+nm[3]+d+nm[5]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl4l[b]==0)&&(rw1l[b]==0)&&(pd4l[b]==0)&&(pb4l[b]==0)){
             if((cl2l[nb]==0)&&(rw5l[nb]==0)&&(pd3l[nb]==0)&&(pb1l[nb]==0)){
               nm[4]=d; nm[22]=nd; uflg[d]=1; uflg[nd]=1;
               cl4h[a]=1; rw1h[a]=1; pd4h[a]=1; pb4h[a]=1;
               cl4l[b]=1; rw1l[b]=1; pd4l[b]=1; pb4l[b]=1;
               cl2h[na]=1; rw5h[na]=1; pd3h[na]=1; pb1h[na]=1;
               cl2l[nb]=1; rw5l[nb]=1; pd3l[nb]=1; pb1l[nb]=1;
               stp08();
               cl2h[na]=0; rw5h[na]=0; pd3h[na]=0; pb1h[na]=0;
               cl2l[nb]=0; rw5l[nb]=0; pd3l[nb]=0; pb1l[nb]=0;
               cl4h[a]=0; rw1h[a]=0; pd4h[a]=0; pb4h[a]=0;
               cl4l[b]=0; rw1l[b]=0; pd4l[b]=0; pb4l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n6 & n20 & n6<n2 *
void stp08(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl1h[a]==0)&&(rw2h[a]==0)&&(pd5h[a]==0)&&(pb2h[a]==0)){
     if((cl5h[na]==0)&&(rw4h[na]==0)&&(pd2h[na]==0)&&(pb3h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((d<nm[2])&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl1l[b]==0)&&(rw2l[b]==0)&&(pd5l[b]==0)&&(pb2l[b]==0)){
             if((cl5l[nb]==0)&&(rw4l[nb]==0)&&(pd2l[nb]==0)&&(pb3l[nb]==0)){
               nm[6]=d; nm[20]=nd; uflg[d]=1; uflg[nd]=1;
               cl1h[a]=1; rw2h[a]=1; pd5h[a]=1; pb2h[a]=1;
               cl1l[b]=1; rw2l[b]=1; pd5l[b]=1; pb2l[b]=1;
               cl5h[na]=1; rw4h[na]=1; pd2h[na]=1; pb3h[na]=1;
               cl5l[nb]=1; rw4l[nb]=1; pd2l[nb]=1; pb3l[nb]=1;
               stp09();
               cl5h[na]=0; rw4h[na]=0; pd2h[na]=0; pb3h[na]=0;
               cl5l[nb]=0; rw4l[nb]=0; pd2l[nb]=0; pb3l[nb]=0;
               cl1h[a]=0; rw2h[a]=0; pd5h[a]=0; pb2h[a]=0;
               cl1l[b]=0; rw2l[b]=0; pd5l[b]=0; pb2l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n11=LSM-n3-n7-n20-n24 & n15 *
void stp09(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl1h[a]==0)&&(rw3h[a]==0)&&(pd4h[a]==0)&&(pb3h[a]==0)){
     if((cl5h[na]==0)&&(rw3h[na]==0)&&(pd3h[na]==0)&&(pb2h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[3]+nm[7]+d+nm[20]+nm[24]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl1l[b]==0)&&(rw3l[b]==0)&&(pd4l[b]==0)&&(pb3l[b]==0)){
             if((cl5l[nb]==0)&&(rw3l[nb]==0)&&(pd3l[nb]==0)&&(pb2l[nb]==0)){
               nm[11]=d; nm[15]=nd; uflg[d]=1; uflg[nd]=1;
```

```
                cl1h[a]=1; rw3h[a]=1; pd4h[a]=1; pb3h[a]=1;
                cl1l[b]=1; rw3l[b]=1; pd4l[b]=1; pb3l[b]=1;
                cl5h[na]=1; rw3h[na]=1; pd3h[na]=1; pb2h[na]=1;
                cl5l[nb]=1; rw3l[nb]=1; pd3l[nb]=1; pb2l[nb]=1;
                stp10();
                cl5h[na]=0; rw3h[na]=0; pd3h[na]=0; pb2h[na]=0;
                cl5l[nb]=0; rw3l[nb]=0; pd3l[nb]=0; pb2l[nb]=0;
                cl1h[a]=0; rw3h[a]=0; pd4h[a]=0; pb3h[a]=0;
                cl1l[b]=0; rw3l[b]=0; pd4l[b]=0; pb3l[b]=0;
                uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n16=LSM-n1-n6-n11-n21 & n10 *
void stp10(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl1h[a]==0)&&(rw4h[a]==0)&&(pd3h[a]==0)&&(pb4h[a]==0)){
     if((cl5h[na]==0)&&(rw2h[na]==0)&&(pd4h[na]==0)&&(pb1h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[1]+nm[6]+nm[11]+d+nm[21]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl1l[b]==0)&&(rw4l[b]==0)&&(pd3l[b]==0)&&(pb4l[b]==0)){
             if((cl5l[nb]==0)&&(rw2l[nb]==0)&&(pd4l[nb]==0)&&(pb1l[nb]==0)){
               nm[16]=d; nm[10]=nd; uflg[d]=1; uflg[nd]=1;
               cl1h[a]=1; rw4h[a]=1; pd3h[a]=1; pb4h[a]=1;
               cl1l[b]=1; rw4l[b]=1; pd3l[b]=1; pb4l[b]=1;
               cl5h[na]=1; rw2h[na]=1; pd4h[na]=1; pb1h[na]=1;
               cl5l[nb]=1; rw2l[nb]=1; pd4l[nb]=1; pb1l[nb]=1;
               stp11();
               cl5h[na]=0; rw2h[na]=0; pd4h[na]=0; pb1h[na]=0;
               cl5l[nb]=0; rw2l[nb]=0; pd4l[nb]=0; pb1l[nb]=0;
               cl1h[a]=0; rw4h[a]=0; pd3h[a]=0; pb4h[a]=0;
               cl1l[b]=0; rw4l[b]=0; pd3l[b]=0; pb4l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n8=LSM-n6-n7-n9-n10 & n18 *
void stp11(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
    if((cl3h[a]==0)&&(rw2h[a]==0)&&(pd2h[a]==0)&&(pb4h[a]==0)){
     if((cl3h[na]==0)&&(rw4h[na]==0)&&(pd5h[na]==0)&&(pb1h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[6]+nm[7]+d+nm[9]+nm[10]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl3l[b]==0)&&(rw2l[b]==0)&&(pd2l[b]==0)&&(pb4l[b]==0)){
             if((cl3l[nb]==0)&&(rw4l[nb]==0)&&(pd5l[nb]==0)&&(pb1l[nb]==0)){
               nm[8]=d; nm[18]=nd; uflg[d]=1; uflg[nd]=1;
               cl3h[a]=1; rw2h[a]=1; pd2h[a]=1; pb4h[a]=1;
               cl3l[b]=1; rw2l[b]=1; pd2l[b]=1; pb4l[b]=1;
               cl3h[na]=1; rw4h[na]=1; pd5h[na]=1; pb1h[na]=1;
               cl3l[nb]=1; rw4l[nb]=1; pd5l[nb]=1; pb1l[nb]=1;
               stp12();
               cl3h[na]=0; rw4h[na]=0; pd5h[na]=0; pb1h[na]=0;
               cl3l[nb]=0; rw4l[nb]=0; pd5l[nb]=0; pb1l[nb]=0;
               cl3h[a]=0; rw2h[a]=0; pd2h[a]=0; pb4h[a]=0;
               cl3l[b]=0; rw2l[b]=0; pd2l[b]=0; pb4l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
        }}}
  }
}
//* Set n12=LSM-n2-n7-n17-n22 & n14 *
void stp12(){
 short a,na,b,nb,d,nd;
  for(a=0;a<5;a++){na=4-a;
```

```
    if((cl2h[a]==0)&&(rw3h[a]==0)&&(pd5h[a]==0)&&(pb4h[a]==0)){
     if((cl4h[na]==0)&&(rw3h[na]==0)&&(pd2h[na]==0)&&(pb1h[na]==0)){
       for(b=0;b<5;b++){nb=4-b; d=a*5+b; nd=CC-d;
         if((nm[2]+nm[7]+d+nm[17]+nm[22]==LSM)&&(uflg[d]==0)&&(uflg[nd]==0)){
           if((cl2l[b]==0)&&(rw3l[b]==0)&&(pd5l[b]==0)&&(pb4l[b]==0)){
             if((cl4l[nb]==0)&&(rw3l[nb]==0)&&(pd2l[nb]==0)&&(pb1l[nb]==0)){
               nm[12]=d; nm[14]=nd; uflg[d]=1; uflg[nd]=1;
               cl2h[a]=1; rw3h[a]=1; pd5h[a]=1; pb4h[a]=1;
               cl2l[b]=1; rw3l[b]=1; pd5l[b]=1; pb4l[b]=1;
               cl4h[na]=1; rw3h[na]=1; pd2h[na]=1; pb1h[na]=1;
               cl4l[nb]=1; rw3l[nb]=1; pd2l[nb]=1; pb1l[nb]=1;
               ansprint();
               cl4h[na]=0; rw3h[na]=0; pd2h[na]=0; pb1h[na]=0;
               cl4l[nb]=0; rw3l[nb]=0; pd2l[nb]=0; pb1l[nb]=0;
               cl2h[a]=0; rw3h[a]=0; pd5h[a]=0; pb4h[a]=0;
               cl2l[b]=0; rw3l[b]=0; pd5l[b]=0; pb4l[b]=0;
               uflg[nd]=0; uflg[d]=0;}}}
       }}}
  }
}
//
//* Print the Answers *
void ansprint(){
 short n;
  cnt++; if(nm[1]==1){cnt1++;}
  anm[cnt3*2][0]=cnt;
  for(n=1;n<26;n++){anm[cnt3*2][n]=nm[n]+1; anm[cnt3*2+1][n]=nm[n];}
  cnt3++; if(cnt3==2){prans(); cnt3=0;}
}
//
//* Print Two Answers *
void prans(){
 short l,l5,n;
  printf(" /D5i   H/         L/%15d#    /D5i   H/         L/%15d#\n",
         anm[0][0],anm[2][0]);
  for(l=0;l<5;l++){l5=l*5;
    for(n=1;n<6;n++){printf("%2d",(anm[1][l5+n])/5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%2d",(anm[1][l5+n])%5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%3d",anm[0][l5+n]);}
    printf("   ");
    for(n=1;n<6;n++){printf("%2d",(anm[3][l5+n])/5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%2d",(anm[3][l5+n])%5);}
    printf(" ");
    for(n=1;n<6;n++){printf("%3d",anm[2][l5+n]);}
    printf("\n");
  }
}
//
```

I am afraid it may look more complicated, but each procedure is written in such a similar way to the others that you can copy one and modify it a little to make another.

#8. Result List of My Recent Computation

Here you are at the list of 'Complete Euler Squares' of Order 5 for the Simultaneous Type. Let me show you the full version of 16 solutions reconstructed.

```
** 'Complete Euler Squares' of Order 5:
  Simultaneous Type: Both Self-Complementary and Pan-Diagonal **
** List of the 16 Standard Solutions **
 [Normalizing Inequality Conditions: n1<n25, n1<n5, n1<21 and n2>n6;]
```

```
 /D5i   H/         L/              1#    /D5i   H/         L/              2#
 0 4 3 2 1  0 2 4 1 3   1 23 20 12  9    0 4 3 2 1  0 2 4 3 1   1 23 20 14  7
 2 1 0 4 3  4 1 3 0 2  15  7  4 21 18    2 1 0 4 3  4 3 1 0 2  15  9  2 21 18
 4 3 2 1 0  3 0 2 4 1  24 16 13 10  2    4 3 2 1 0  1 0 2 4 3  22 16 13 10  4
 1 0 4 3 2  2 4 1 3 0   8  5 22 19 11    1 0 4 3 2  2 4 3 1 0   8  5 24 17 11
 3 2 1 0 4  1 3 0 2 4  17 14  6  3 25    3 2 1 0 4  3 1 0 2 4  19 12  6  3 25
 /D5i   H/         L/              3#    /D5i   H/         L/              4#
 0 4 1 2 3  0 2 4 1 3   1 23 10 12 19    0 4 1 2 3  0 2 4 3 1   1 23 10 14 17
 2 3 0 4 1  4 1 3 0 2  15 17  4 21  8    2 3 0 4 1  4 3 1 0 2  15 19  2 21  8
 4 1 2 3 0  3 0 2 4 1  24  6 13 20  2    4 1 2 3 0  1 0 2 4 3  22  6 13 20  4
 3 0 4 1 2  2 4 1 3 0  18  5 22  9 11    3 0 4 1 2  2 4 3 1 0  18  5 24  7 11
 1 2 3 0 4  1 3 0 2 4   7 14 16  3 25    1 2 3 0 4  3 1 0 2 4   9 12 16  3 25
 /D5i   H/         L/              5#    /D5i   H/         L/              6#
 0 4 3 2 1  1 2 3 0 4   2 23 19 11 10    0 4 3 2 1  1 2 3 4 0   2 23 19 15  6
 2 1 0 4 3  3 0 4 1 2  14  6  5 22 18    2 1 0 4 3  3 4 0 1 2  14 10  1 22 18
 4 3 2 1 0  4 1 2 3 0  25 17 13  9  1    4 3 2 1 0  0 1 2 3 4  21 17 13  9  5
 1 0 4 3 2  2 3 0 4 1   8  4 21 20 12    1 0 4 3 2  2 3 4 0 1   8  4 25 16 12
 3 2 1 0 4  0 4 1 2 3  16 15  7  3 24    3 2 1 0 4  4 0 1 2 3  20 11  7  3 24
 /D5i   H/         L/              7#    /D5i   H/         L/              8#
 0 4 1 2 3  1 2 3 0 4   2 23  9 11 20    0 4 1 2 3  1 2 3 4 0   2 23  9 15 16
 2 3 0 4 1  3 0 4 1 2  14 16  5 22  8    2 3 0 4 1  3 4 0 1 2  14 20  1 22  8
 4 1 2 3 0  4 1 2 3 0  25  7 13 19  1    4 1 2 3 0  0 1 2 3 4  21  7 13 19  5
 3 0 4 1 2  2 3 0 4 1  18  4 21 10 12    3 0 4 1 2  2 3 4 0 1  18  4 25  6 12
 1 2 3 0 4  0 4 1 2 3   6 15 17  3 24    1 2 3 0 4  4 0 1 2 3  10 11 17  3 24
 /D5i   H/         L/              9#    /D5i   H/         L/             10#
 0 4 3 2 1  3 2 1 0 4   4 23 17 11 10    0 4 3 2 1  3 2 1 4 0   4 23 17 15  6
 2 1 0 4 3  1 0 4 3 2  12  6  5 24 18    2 1 0 4 3  1 4 0 3 2  12 10  1 24 18
 4 3 2 1 0  4 3 2 1 0  25 19 13  7  1    4 3 2 1 0  0 3 2 1 4  21 19 13  7  5
 1 0 4 3 2  2 1 0 4 3   8  2 21 20 14    1 0 4 3 2  2 1 4 0 3   8  2 25 16 14
 3 2 1 0 4  0 4 3 2 1  16 15  9  3 22    3 2 1 0 4  4 0 3 2 1  20 11  9  3 22
 /D5i   H/         L/             11#    /D5i   H/         L/             12#
 0 4 1 2 3  3 2 1 0 4   4 23  7 11 20    0 4 1 2 3  3 2 1 4 0   4 23  7 15 16
 2 3 0 4 1  1 0 4 3 2  12 16  5 24  8    2 3 0 4 1  1 4 0 3 2  12 20  1 24  8
 4 1 2 3 0  4 3 2 1 0  25  9 13 17  1    4 1 2 3 0  0 3 2 1 4  21  9 13 17  5
 3 0 4 1 2  2 1 0 4 3  18  2 21 10 14    3 0 4 1 2  2 1 4 0 3  18  2 25  6 14
 1 2 3 0 4  0 4 3 2 1   6 15 19  3 22    1 2 3 0 4  4 0 3 2 1  10 11 19  3 22
 /D5i   H/         L/             13#    /D5i   H/         L/             14#
 0 4 3 2 1  4 2 0 1 3   5 23 16 12  9    0 4 3 2 1  4 2 0 3 1   5 23 16 14  7
 2 1 0 4 3  0 1 3 4 2  11  7  4 25 18    2 1 0 4 3  0 3 1 4 2  11  9  2 25 18
 4 3 2 1 0  3 4 2 0 1  24 20 13  6  2    4 3 2 1 0  1 4 2 0 3  22 20 13  6  4
 1 0 4 3 2  2 0 1 3 4   8  1 22 19 15    1 0 4 3 2  2 0 3 1 4   8  1 24 17 15
 3 2 1 0 4  1 3 4 2 0  17 14 10  3 21    3 2 1 0 4  3 1 4 2 0  19 12 10  3 21
 /D5i   H/         L/             15#    /D5i   H/         L/             16#
 0 4 1 2 3  4 2 0 1 3   5 23  6 12 19    0 4 1 2 3  4 2 0 3 1   5 23  6 14 17
 2 3 0 4 1  0 1 3 4 2  11 17  4 25  8    2 3 0 4 1  0 3 1 4 2  11 19  2 25  8
 4 1 2 3 0  3 4 2 0 1  24 10 13 16  2    4 1 2 3 0  1 4 2 0 3  22 10 13 16  4
 3 0 4 1 2  2 0 1 3 4  18  1 22  9 15    3 0 4 1 2  2 0 3 1 4  18  1 24  7 15
 1 2 3 0 4  1 3 4 2 0   7 14 20  3 21    1 2 3 0 4  3 1 4 2 0   9 12 20  3 21

  [Total Solution Counts(n1=1/All) = 4/16]  OK!

** Calculated and Listed by Kanji Setsuda on
    Feb.14, 2015 with MacOSX 10.10.2 and Xcode 6.1.1 **
```

All of these results are just the same with those 16 "Suzuki Squares": Simultaneous Magic Squares 5x5: Self-complementary and Pan-diagonal;

It does not only mean this program could work all right, but also means it could verify completely that all of Suzuki Squares 5x5 are really "Complete Euler Squares".

How happy I am to know that!

#9. 'Euler Squares' 5x5 of Self-Complementary Magic Type

How about Self-complementary type? Can we make any 'Complete Euler Square' for that type? I mean 'Non-Simultaneous' type and pure Self-complementary MS55.

No, we can't.

According to our definition we call 'C.E.S.' for what has all pan-diagonals made up of {0, 1, 2, 3 and 4} using each strictly once. But non-Simultaneous type does not always have to take the constant sum of all pan-diagonals.

It is the problem of definition.

If you want to have any 'Euler Square' 5x5 of Self-complementary type, you should assume the 'Latin Structure' only for every column and every row.

The next lists show the contrast of those definitions of 'Euler Squares' of Order 5.

** **List 1:** The 32 Standard Solutions of the Self-Complementary MS55
Made by the 5-th Increment Number System and Greco-Latinian Method **
(Used_Unit:H/L/; Sol_Numb#; Sum_Checks:\Diag1/Diag2|Row|Column|)

```
   3/    5/              1#\65/65|      4/    6/              2#\65/65|
04321 02413   1 23 20 12  9|65|65|   03421 02143   1 18 22 15  9|65|65|
21043 41302  15  7  4 21 18|65|65|   21340 31402  14  7 20 21  3|65|65|
43210 30241  24 16 13 10  2|65|65|   14203 43210  10 24 13  2 16|65|65|
10432 24130   8  5 22 19 11|65|65|   40132 24031  23  5  6 19 12|65|65|
32104 13024  17 14  6  3 25|65|65|   32014 10324  17 11  4  8 25|65|65|
   3/    9/              3#\65/65|      4/   10/              4#\65/65|
04321 02431   1 23 20 14  7|65|65|   03421 02341   1 18 24 15  7|65|65|
21043 43102  15  9  2 21 18|65|65|   21340 13402  12  9 20 21  3|65|65|
43210 10243  22 16 13 10  4|65|65|   14203 41230  10 22 13  4 16|65|65|
10432 24310   8  5 24 17 11|65|65|   40132 24013  23  5  6 17 14|65|65|
32104 31024  19 12  6  3 25|65|65|   32014 30124  19 11  2  8 25|65|65|
  15/    5/              5#\65/65|     10/    4/              6#\65/65|
04123 02413   1 23 10 12 19|65|65|   02341 03421   1 14 20 23  7|65|65|
23041 41302  15 17  4 21  8|65|65|   13402 21340   8 17 24  5 11|65|65|
41230 30241  24  6 13 20  2|65|65|   41230 14203  22 10 13 16  4|65|65|
30412 24130  18  5 22  9 11|65|65|   24013 40132  15 21  2  9 18|65|65|
12304 13024   7 14 16  3 25|65|65|   30124 32014  19  3  6 12 25|65|65|
  15/    9/              7#\65/65|     10/   16/              8#\65/65|
04123 02431   1 23 10 14 17|65|65|   02341 01423   1 12 20 23  9|65|65|
23041 43102  15 19  2 21  8|65|65|   13402 23140   8 19 22  5 11|65|65|
41230 10243  22  6 13 20  4|65|65|   41230 34201  24 10 13 16  2|65|65|
30412 24310  18  5 24  7 11|65|65|   24013 40312  15 21  4  7 18|65|65|
12304 31024   9 12 16  3 25|65|65|   30124 12034  17  3  6 14 25|65|65|
  3/ 21/    9#     4/ 22/   10#     3/ 26/   11#     4/ 25/   12#
 2 23 19 11 10    2 18 21 14 10    2 23 19 15  6    2 18 25 14  6
14  6  5 22 18   15  6 19 22  3   14 10  1 22 18   11 10 19 22  3
25 17 13  9  1    9 25 13  1 17   21 17 13  9  5    9 21 13  5 17
 8  4 21 20 12   23  4  7 20 11    8  4 25 16 12   23  4  7 16 15
16 15  7  3 24   16 12  5  8 24   20 11  7  3 24   20 12  1  8 24
 15/ 21/   13#    10/ 19/   14#    15/ 26/   15#    10/ 32/   16#
 2 23  9 11 20    2 15 19 23  6    2 23  9 15 16    2 11 19 23 10
14 16  5 22  8    8 16 25  4 12   14 20  1 22  8    8 20 21  4 12
25  7 13 19  1   21  9 13 17  5   21  7 13 19  5   25  9 13 17  1
18  4 21 10 12   14 22  1 10 18   18  4 25  6 12   14 22  5  6 18
 6 15 17  3 24   20  3  7 11 24   10 11 17  3 24   16  3  7 15 24
  3/ 39/   17#     4/ 40/   18#     3/ 44/   19#     4/ 43/   20#
 4 23 17 11 10    4 18 21 12 10    4 23 17 15  6    4 18 25 12  6
12  6  5 24 18   15  6 17 24  3   12 10  1 24 18   11 10 17 24  3
25 19 13  7  1    7 25 13  1 19   21 19 13  7  5    7 21 13  5 19
 8  2 21 20 14   23  2  9 20 11    8  2 25 16 14   23  2  9 16 15
16 15  9  3 22   16 14  5  8 22   20 11  9  3 22   20 14  1  8 22
```

```
  15/ 39/   21#    10/ 33/   22#    15/ 44/   23#    10/ 46/   24#
 4 23  7 11 20    4 15 17 23  6    4 23  7 15 16    4 11 17 23 10
12 16  5 24  8    8 16 25  2 14   12 20  1 24  8    8 20 21  2 14
25  9 13 17  1   21  7 13 19  5   21  9 13 17  5   25  7 13 19  1
18  2 21 10 14   12 24  1 10 18   18  2 25  6 14   12 24  5  6 18
 6 15 19  3 22   20  3  9 11 22   10 11 19  3 22   16  3  9 15 22
   3/ 56/   25#     4/ 55/   26#     3/ 60/   27#     4/ 59/   28#
 5 23 16 12  9    5 18 22 11  9    5 23 16 14  7    5 18 24 11  7
11  7  4 25 18   14  7 16 25  3   11  9  2 25 18   12  9 16 25  3
24 20 13  6  2    6 24 13  2 20   22 20 13  6  4    6 22 13  4 20
 8  1 22 19 15   23  1 10 19 12    8  1 24 17 15   23  1 10 17 14
17 14 10  3 21   17 15  4  8 21   19 12 10  3 21   19 15  2  8 21
  15/ 56/   29#    10/ 49/   30#    15/ 60/   31#    10/ 61/   32#
 5 23  6 12 19    5 14 16 23  7    5 23  6 14 17    5 12 16 23  9
11 17  4 25  8    8 17 24  1 15   11 19  2 25  8    8 19 22  1 15
24 10 13 16  2   22  6 13 20  4   22 10 13 16  4   24  6 13 20  2
18  1 22  9 15   11 25  2  9 18   18  1 24  7 15   11 25  4  7 18
 7 14 20  3 21   19  3 10 12 21    9 12 20  3 21   17  3 10 14 21
[Total Counts of Solutions(n1=1/All) = 8/32]  OK!

** Calculated and Listed by Kanji Setsuda on Jun.27, 2010;
    Recompiled on Feb.14, 2015 with MacOSX 10.10.2 and Xcode 6.1.1; **
```

Each of these 32 solutions has its two primary diagonals made of 'Latin Units'.

```
** List 2: 'Euler Squares 5x5' of Self-Complementary Magic Type: **
  ** Without any Definition about their two Primary Diagonals **
            1/  /D5i                              5/  /D5i
 1 20 22 14  8  0 3 4 2 1  0 4 1 3 2    1 18 24 15  7  0 3 4 2 1  0 2 3 4 1
24  3 10 17 11  4 0 1 3 2  3 2 4 1 0   23  5  6 17 14  4 0 1 3 2  2 4 0 1 3
 7 21 13  5 19  1 4 2 0 3  1 0 2 4 3   10 22 13  4 16  1 4 2 0 3  4 1 2 3 0
15  9 16 23  2  2 1 3 4 0  4 3 0 2 1   12  9 20 21  3  2 1 3 4 0  1 3 4 0 2
18 12  4  6 25  3 2 0 1 4  2 1 3 0 4   19 11  2  8 25  3 2 0 1 4  3 0 1 2 4
            9/  /D5i                             15/  /D5i
 1 20 24 12  8  0 3 4 2 1  0 4 3 1 2    1 18 24 15  7  0 3 4 2 1  0 2 3 4 1
15  7 16 23  4  2 1 3 4 0  4 1 0 2 3   12  9 20 21  3  2 1 3 4 0  1 3 4 0 2
 9 21 13  5 17  1 4 2 0 3  3 0 2 4 1   10 22 13  4 16  1 4 2 0 3  4 1 2 3 0
22  3 10 19 11  4 0 1 3 2  1 2 4 3 0   23  5  6 17 14  4 0 1 3 2  2 4 0 1 3
18 14  2  6 25  3 2 0 1 4  2 3 1 0 4   19 11  2  8 25  3 2 0 1 4  3 0 1 2 4
           21/  /D5i                             25/  /D5i
 1 10 19 23 12  0 1 3 4 2  0 4 3 2 1    1  8 20 24 12  0 1 3 4 2  0 2 4 3 1
22 11  5  9 18  4 2 0 1 3  1 0 4 3 2   23 15  4  7 16  4 2 0 1 3  2 4 3 1 0
20 24 13  2  6  3 4 2 0 1  4 3 2 1 0   17 21 13  5  9  3 4 2 0 1  1 0 2 4 3
 8 17 21 15  4  1 3 4 2 0  2 1 0 4 3   10 19 22 11  3  1 3 4 2 0  4 3 1 0 2
14  3  7 16 25  2 0 1 3 4  3 2 1 0 4   14  2  6 18 25  2 0 1 3 4  3 1 0 2 4
           29/  /D5i                             35/  /D5i
 1 14 20 23  7  0 2 3 4 1  0 3 4 2 1    1 12 24 20  8  0 2 4 3 1  0 1 3 4 2
 8 17 24  5 11  1 3 4 0 2  2 1 3 4 0   22 19 10  3 11  4 3 1 0 2  1 3 4 2 0
22 10 13 16  4  4 1 2 3 0  1 4 2 0 3    9  5 13 21 17  1 0 2 4 3  3 4 2 0 1
15 21  2  9 18  2 4 0 1 3  4 0 1 3 2   15 23 16  7  4  2 4 3 1 0  4 2 0 1 3
19  3  6 12 25  3 0 1 2 4  3 2 0 1 4   18  6  2 14 25  3 1 0 2 4  2 0 1 3 4
           41/  /D5i                             45/  /D5i
 1 14 20 23  7  0 2 3 4 1  0 3 4 2 1    1 12 24 20  8  0 2 4 3 1  0 1 3 4 2
15 21  2  9 18  2 4 0 1 3  4 0 1 3 2   15 23 16  7  4  2 4 3 1 0  4 2 0 1 3
22 10 13 16  4  4 1 2 3 0  1 4 2 0 3    9  5 13 21 17  1 0 2 4 3  3 4 2 0 1
 8 17 24  5 11  1 3 4 0 2  2 1 3 4 0   22 19 10  3 11  4 3 1 0 2  1 3 4 2 0
19  3  6 12 25  3 0 1 2 4  3 2 0 1 4   18  6  2 14 25  3 1 0 2 4  2 0 1 3 4
```

```
           49/  /D5i                          53/  /D5i
 2 19 21 15  8  0 3 4 2 1  1 3 0 4 2    2 18 25 14  6  0 3 4 2 1  1 2 4 3 0
25  3  9 16 12  4 0 1 3 2  4 2 3 0 1   23  4  7 16 15  4 0 1 3 2  2 3 1 0 4
 6 22 13  4 20  1 4 2 0 3  0 1 2 3 4    9 21 13  5 17  1 4 2 0 3  3 0 2 4 1
14 10 17 23  1  2 1 3 4 0  3 4 1 2 0   11 10 19 22  3  2 1 3 4 0  0 4 3 1 2
18 11  5  7 24  3 2 0 1 4  2 0 4 1 3   20 12  1  8 24  3 2 0 1 4  4 1 0 2 3
           57/  /D5i                          63/  /D5i
 2 19 25 11  8  0 3 4 2 1  1 3 4 0 2    2 18 25 14  6  0 3 4 2 1  1 2 4 3 0
14  6 17 23  5  2 1 3 4 0  3 0 1 2 4   11 10 19 22  3  2 1 3 4 0  0 4 3 1 2
10 22 13  4 16  1 4 2 0 3  4 1 2 3 0    9 21 13  5 17  1 4 2 0 3  3 0 2 4 1
21  3  9 20 12  4 0 1 3 2  0 2 3 4 1   23  4  7 16 15  4 0 1 3 2  2 3 1 0 4
18 15  1  7 24  3 2 0 1 4  2 4 0 1 3   20 12  1  8 24  3 2 0 1 4  4 1 0 2 3
           69/  /D5i                          73/  /D5i
 2  9 20 23 11  0 1 3 4 2  1 3 4 2 0    2  8 19 25 11  0 1 3 4 2  1 2 3 4 0
21 12  4 10 18  4 2 0 1 3  0 1 3 4 2   23 14  5  6 17  4 2 0 1 3  2 3 4 0 1
19 25 13  1  7  3 4 2 0 1  3 4 2 0 1   16 22 13  4 10  3 4 2 0 1  0 1 2 3 4
 8 16 22 14  5  1 3 4 2 0  2 0 1 3 4    9 20 21 12  3  1 3 4 2 0  3 4 0 1 2
15  3  6 17 24  2 0 1 3 4  4 2 0 1 3   15  1  7 18 24  2 0 1 3 4  4 0 1 2 3
           77/  /D5i                          83/  /D5i
 2 14 25 18  6  0 2 4 3 1  1 3 4 2 0    2 11 25 19  8  0 2 4 3 1  1 0 4 3 2
23 16  7  4 15  4 3 1 0 2  2 0 1 3 4   21 20  9  3 12  4 3 1 0 2  0 4 3 2 1
 9  5 13 21 17  1 0 2 4 3  3 4 2 0 1   10  4 13 22 16  1 0 2 4 3  4 3 2 1 0
11 22 19 10  3  2 4 3 1 0  0 1 3 4 2   14 23 17  6  5  2 4 3 1 0  3 2 1 0 4
20  8  1 12 24  3 1 0 2 4  4 2 0 1 3   18  7  1 15 24  3 1 0 2 4  2 1 0 4 3
           89/  /D5i                          93/  /D5i
 2 14 25 18  6  0 2 4 3 1  1 3 4 2 0    2 11 25 19  8  0 2 4 3 1  1 0 4 3 2
11 22 19 10  3  2 4 3 1 0  0 1 3 4 2   14 23 17  6  5  2 4 3 1 0  3 2 1 0 4
 9  5 13 21 17  1 0 2 4 3  3 4 2 0 1   10  4 13 22 16  1 0 2 4 3  4 3 2 1 0
23 16  7  4 15  4 3 1 0 2  2 0 1 3 4   21 20  9  3 12  4 3 1 0 2  0 4 3 2 1
20  8  1 12 24  3 1 0 2 4  4 2 0 1 3   18  7  1 15 24  3 1 0 2 4  2 1 0 4 3
           97/  /D5i                         101/  /D5i
 3 20 21 14  7  0 3 4 2 1  2 4 0 3 1    3 19 22 15  6  0 3 4 2 1  2 3 1 4 0
22  1  9 18 15  4 0 1 3 2  1 0 3 2 4   21  2 10 18 14  4 0 1 3 2  0 1 4 2 3
10 24 13  2 16  1 4 2 0 3  4 3 2 1 0    9 25 13  1 17  1 4 2 0 3  3 4 2 0 1
11  8 17 25  4  2 1 3 4 0  0 2 1 4 3   12  8 16 24  5  2 1 3 4 0  1 2 0 3 4
19 12  5  6 23  3 2 0 1 4  3 1 4 0 2   20 11  4  7 23  3 2 0 1 4  4 0 3 1 2

 . . . .(Skip). . . .

           365/  /D5i                         367/  /D5i
 9  2 25 18 11  1 0 4 3 2  3 1 4 2 0    9  3 22 20 11  1 0 4 3 2  3 2 1 4 0
 3 21 19 12 10  0 4 3 2 1  2 0 3 1 4    2 25 16 14  8  0 4 3 2 1  1 4 0 3 2
22 20 13  6  4  4 3 2 1 0  1 4 2 0 3   21 19 13  7  5  4 3 2 1 0  0 3 2 1 4
16 14  7  5 23  3 2 1 0 4  0 3 1 4 2   18 12 10  1 24  3 2 1 0 4  2 1 4 0 3
15  8  1 24 17  2 1 0 4 3  4 2 0 3 1   15  6  4 23 17  2 1 0 4 3  4 0 3 2 1
           369/  /D5i                         371/  /D5i
10 19 21  3 12  1 3 4 0 2  4 3 0 2 1   10 18 24  1 12  1 3 4 0 2  4 2 3 0 1
18  2  9 11 25  3 0 1 2 4  2 1 3 0 4   17  4  6 15 23  3 0 1 2 4  1 3 0 4 2
22  6 13 20  4  4 1 2 3 0  1 0 2 4 3   21  7 13 19  5  4 1 2 3 0  0 1 2 3 4
 1 15 17 24  8  0 2 3 4 1  0 4 1 3 2    3 11 20 22  9  0 2 3 4 1  2 0 4 1 3
14 23  5  7 16  2 4 0 1 3  3 2 4 1 0   14 25  2  8 16  2 4 0 1 3  3 4 1 2 0
           373/  /D5i                         377/  /D5i
10  3 21 19 12  1 0 4 3 2  4 2 0 3 1   10  1 24 18 12  1 0 4 3 2  4 0 3 2 1
18 11  9  2 25  3 2 1 0 4  2 0 3 1 4   17 15  6  4 23  3 2 1 0 4  1 4 0 3 2
22 20 13  6  4  4 3 2 1 0  1 4 2 0 3   21 19 13  7  5  4 3 2 1 0  0 3 2 1 4
 1 24 17 15  8  0 4 3 2 1  0 3 1 4 2    3 22 20 11  9  0 4 3 2 1  2 1 4 0 3
14  7  5 23 16  2 1 0 4 3  3 1 4 2 0   14  8  2 25 16  2 1 0 4 3  3 2 1 4 0
```

```
          381/  /D5i                          383/  /D5i
10  1 24 18 12  1 0 4 3 2  4 0 3 2 1   10  3 21 19 12  1 0 4 3 2  4 2 0 3 1
 3 22 20 11  9  0 4 3 2 1  2 1 4 0 3    1 24 17 15  8  0 4 3 2 1  0 3 1 4 2
21 19 13  7  5  4 3 2 1 0  0 3 2 1 4   22 20 13  6  4  4 3 2 1 0  1 4 2 0 3
17 15  6  4 23  3 2 1 0 4  1 4 0 3 2   18 11  9  2 25  3 2 1 0 4  2 0 3 1 4
14  8  2 25 16  2 1 0 4 3  3 2 1 4 0   14  7  5 23 16  2 1 0 4 3  3 1 4 2 0
[Count(n1=1/All) = 48/384]
```

We could find only a few solutions as many as 384 among the 48544 SCMS55.
I haven't yet known anything important about this, while I have been long thinking.
But don't you think it is nice we could build 'Euler Squares' even for S-C type?

#10. Comment

This is the first successful result of my elaborate computation to try to make any solution set of "Complete Euler Squares". Though it might look too complicated, I am proud of my experience I could surely have got some skills to solve the problem and have any special key in my hand to open the door to the new world.

I am lucky I could have invented another two ways of composing Complete Euler Squares: "Compositions by Knight's Tour", and "New Euler's Method". I want you to read my articles I explained about them in the following sections.

Those new methods can do our job very fast, faster than anything else.

Let's go on to the next challenge to build Complete Euler Squares of Order 7.

```
(The Original written in English on March 27, 2003 by Kanji Setsuda;
  Retyped Edition on February 21, 2007 with MacOSX and Xcode2.2;
    Recompiled on February 14, 2015 with MacOSX 10.10.2 & Xcode 6.1.1)
--------------------------------------------------------------------------------
    E-Mail Address <jag12001@nifty.com>
```