**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:

"Positional Number System of the Base N"

# Section 11: Various "Euler Squares" of Order 3, 5 and 7 made by "New Euler's Method" using PNS of Base 3, 5 and 7

## 0. Our purpose

In 2012 I changed my style of thinking a little about "Euler Squares", "Complete Euler Squares" and "New Euler's Method" and re-defined them. I want to make some revision to this article according to those new ideas. But I would keep my original purpose and respect the old contents as much as possible.

I want to examine our **New Euler's Method** using Positional Number System of Base 3, 5 and 7, namely the 3rd, 5th and 7th Increment Number System, by applying them to the cases of order 3, 5 and 7.

I once skipped applying it to the first and smallest case of magic things of order 3.

When I invented and applied the method of "Knight's Tour Composition" for the orders 5 and 7, I did not reach to any total idea of New Euler's Method yet, though these two applications are essentially similar, I would say, almost the same.

Let's examine our new method by applying to those several cases now, shall we?

## 1. How to apply our New Euler's Method to Magic Square 3x3

The New Euler's Method should be explained by the following diagrams of the two way symmetrical processes: Decomposition by the 3-rd Increment Number System and composition by the same system for the objective Magic Squares of Order 3.

```
** New Euler's Method for the Standard Magic Squares of Order 3 **
 * Decomposition by the 3rd Increment Number System *
 [[Classic] [Mathem] /D3i: High| Low|; Sum Checks of Each: \D1/D2|Row|Column|;]
  [*]     \15/15| [Math]  \12/12| /D3i H\3/3|      L\3/3|
   2  9  4|15|15|  1  8  3|12|12|  0 2 1|3|3|  1 2 0|3|3|
   7  5  3|15|15|  6  4  2|12|12|  2 1 0|3|3|  0 1 2|3|3|
   6  1  8|15|15|  5  0  7|12|12|  1 0 2|3|3|  2 0 1|3|3|

 * Composition by the 3rd Increment Number System *
  [/EU: High| Low|; [Mathem] Classic# Sum Checks of Each: \D1/D2|Row|Column|;]
  /EU *H\3/3|    **L\3/3|  [Math]  \12/12|        *#\15/15|
   0 2 1|3|3|  1 2 0|3|3|  1  8  3|12|12|   2  9  4|15|15|
   2 1 0|3|3|  0 1 2|3|3|  6  4  2|12|12|   7  5  3|15|15|
   1 0 2|3|3|  2 0 1|3|3|  5  0  7|12|12|   6  1  8|15|15|

  Calculation: 0 x 3 + 1 = 1;  2 x 3 + 2 = 8;  1 x 3 + 0 = 3;
               2 x 3 + 0 = 6;  1 x 3 + 1 = 4;  0 x 3 + 2 = 2;
               1 x 3 + 2 = 5;  0 x 3 + 0 = 0;  2 x 3 + 1 = 7;
```

Suppose the first upper process of decomposition as a 'function', and then you can think the 'inverse function' of that for the second lower process of composition.

In the upper process the second form in mathematical notation is made after the original square in classical notation, always by subtracting 1 from each element. Divide each number in the [math] by 3, and put the integer part of quotient into the higher unit right and put the remainder into the lower one on the most right.

A pair of those result units on the right hand side are called "Decomposed Units by

the 3rd Increment Number System", or simply "/D3i".

In the lower process above they show the composition in the symmetrical way to the upper case. A pair of these two little units on the left, as you see they have the same contents with the last two upper, are called "**Euler Units**" for the New Euler's Method. We should take the same kind of two components to combine and compose the original object Magic Squares of Order 3 by our arithmetic calculation.

Multiply each number in higher unit by 3 and to the product add each number in the corresponding position of lower. Then you should be able to have the Math form. If you only add 1 to every number of this Math form, you could have reconstructed our object on the whole in the classical notation at last.

New Euler's Method needs real **Euler Units** at first. But how can we get them?

We must avoid direct borrowing from the decomposed results above. It only means 'tautology' and gives us nothing important. With any independent method, we should build that kind of real Euler Units one by one by ourselves. Sincerity and diligence is necessary for us to keep all the way.

What conditions do we have to suppose for defining New Euler Units, then?

We have to learn everything from the original definitions of the objective MS33.

Study the following two sets of definitions. I have made the second set for our Euler Units after the first original set, modifying it only a little:

```
//
//** #1: Definitions for Standard Magic Squares of Order 3 **
//   .--.--.--. * Basic Form and Equations: SC=15 *
//   |n1|n2|n3|    n1+n2+n3=SC ...  sm1;
//   |--+--+--|    n4+n5+n6=SC ...  sm2;
//   |n4|n5|n6|    n7+n8+n9=SC ...  sm3;
//   |--+--+--|    n1+n4+n7=SC ...  sm4;
//   |n7|n8|n9|    n2+n5+n8=SC ...  sm5;
//   '--'--'--'    n3+n6+n9=SC ...  sm6;
//     n1+n5+n9=SC ...  sm7;   n3+n5+n7=SC ...  sm8;
//
//** #2: Definitions for the Euler Units of Order 3 **
//   .--.--.--. * Basic Form and Equations: EC=3 *
//   |n1|n2|n3|    n1+n2+n3=EC ...  esm1;
//   |--+--+--|    n4+n5+n6=EC ...  esm2;
//   |n4|n5|n6|    n7+n8+n9=EC ...  esm3;
//   |--+--+--|    n1+n4+n7=EC ...  esm4;
//   |n7|n8|n9|    n2+n5+n8=EC ...  esm5;
//   '--'--'--'    n3+n6+n9=EC ...  esm6;
//     n1+n5+n9=EC ...  esm7;  n3+n5+n7=EC ...  esm8;
//
```

They are very similar, almost the same, as you see. This 'similar structure' between the whole object and its Euler Units is necessary for our **Euler Squares**.

On top of that, let's add one more important rule to those definitions above:

> "In each Unit any number of {0, 1, 2} must be used just **three times** as often as the others. More or less often it must not be used."

I must mention here I do not take care about the so-called '**Latin Units**' or 'Latin Conditions' at all now. It is because we can make neither "Complete Euler Squares" nor "Greco-Latin Squares" of Order 3. We cannot avoid one of the primary diagonals having the content pattern 1+1+1=3, while all the other rows, columns and another diagonal could really have the only Latin Pattern 0+1+2=3.

Now let's try to build our real Euler Units only with these new definitions above.

Let me show you the making Program I prepared as follows in the next short list.

```
//** Standard Magic Squares of Order 3 Reconstructed by the **
```

```c
//**  3rd Increment Number System and New Euler's Method **
//** 'NEulerMS33S.c': Dictated by Kanji Setsuda on **
//**  Apr.11, 2006, Dec. 9, 2011 and Apr.14, 2012  **
//**  Working with MacOSX 10.7.3 & Xcode 4.3.2  **
//
#include <stdio.h>
//
short cnt, ecnt, cnt3;
short LSM;
short u1,u2;
short nm[10], uflg[10], tn[10];
short anm[5][10];
short teu[5][10];
short mtc[5][5], cmtc[5][5], scmtc[5][5];
short solt[17][12];
short cs[5][9];
//
void stp01(void), stp02(void), stp03(void), stp04(void), stp05(void);
void stp06(void), stp07(void), stp08(void), stp09(void);
void hansrecord(void), eurecord(void);
void preunit(void);
//
void cmbcmp(void), recprans(void), anspr(void);
void chksms(short x);
//
//** Main Program **
int main(void){
 short n;
  printf("\n");
  printf("** Reconstruct Standard Magic Squares of Order 3 by\n");
  printf("    the 3rd Increment Number System and New Euler's Method **\n");
  for(n=0;n<10;n++){nm[n]=0;}
  for(n=0;n<3;n++){uflg[n]=0;}
  cnt=0; cnt3=0; LSM=3;
  printf("\n");
  printf("** List of Euler Units **\n");
  stp01();                      // Make the Euler Units
  ecnt=cnt;
  preunit();                    //* Classify and Print the Euler Units */
  printf("\n");
  printf("** List of Standard Magic Squares of Order 3 Reconstructed **\n");
  printf(" [/Euler Units: High| Low|; [Math] Classic#;\n");
  printf("     with Sum Checks of Each: \\D1/D2|Row|Column|;]\n");
  cnt=0; cnt3=0;
  cmbcmp();                     // Combine, Compose and Print
  printf(" [Solution Counts = %d]  [OK!]\n",cnt);
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on Apr.14, 2012 with MacOSX 10.7.3 & Xcode 4.3.2 **\n");
  printf("\n");
 return 0;
}
//
//* Make the Euler Units
//Level #1:
//Set n1
void stp01(){
 short i;
  for(i=0;i<3;i++){
    if(uflg[i]<3){nm[1]=i;
      uflg[i]++; stp02(); uflg[i]--;}
  }
}
//Set n2
void stp02(){
 short i;
```

```
   for(i=0;i<3;i++){
     if(uflg[i]<3){nm[2]=i;
       uflg[i]++; stp03(); uflg[i]--;}
   }
}
//Set n3=3-n1-n2
void stp03(){
 short i;
   i=LSM-nm[1]-nm[2];
   if((0<=i)&&(i<3)){
     if(uflg[i]<3){nm[3]=i;
      uflg[i]++; stp04(); uflg[i]--;}}
}
//Set n4
void stp04(){
 short i;
   for(i=0;i<3;i++){
     if(uflg[i]<3){nm[4]=i;
       uflg[i]++; stp05(); uflg[i]--;}
   }
}
//Set n5
void stp05(){
 short i;
   for(i=0;i<3;i++){
     if(uflg[i]<3){nm[5]=i;
       uflg[i]++; stp06(); uflg[i]--;}
   }
}
//Set n6=3-n4-n5
void stp06(){
 short i;
   i=LSM-nm[4]-nm[5];
   if((0<=i)&&(i<3)){
     if(uflg[i]<3){nm[6]=i;
       uflg[i]++; stp07(); uflg[i]--;}}
}
//Set n7=3-n1-n4
void stp07(){
 short i,j;
   i=LSM-nm[1]-nm[4];
   j=LSM-nm[3]-nm[5];
   if((0<=i)&&(i<3)&&(i==j)){
     if(uflg[i]<3){nm[7]=i;
       uflg[i]++; stp08(); uflg[i]--;}}
}
/* Set n8=3-n2-n5 */
void stp08(){
 short i;
   i=LSM-nm[2]-nm[5];
   if((0<=i)&&(i<3)){
     if(uflg[i]<3){nm[8]=i;
       uflg[i]++; stp09(); uflg[i]--;}}
}
/* Set n9=3-n7-n8 & n9=3-n3-n6 & n9=3-n1-n5 */
void stp09(){
 short i,j,k;
   i=LSM-nm[7]-nm[8];
   j=LSM-nm[3]-nm[6];
   k=LSM-nm[1]-nm[5];
   if((0<=i)&&(i<3)){
     if((i==j)&&(i==k)){
       if(uflg[i]<3){nm[9]=i;
         uflg[i]++; eurecord(); uflg[i]--;}}}
}
```

```
//
//* Record the Answers *
void eurecord(){
 short n;
  teu[cnt][0]=cnt+1;
  for(n=1;n<10;n++){teu[cnt][n]=nm[n];}
  cnt++;
}
//
//* Classify and Print the Euler Units *
void preunit(){
 short t,l,l3,m,n;
  for(m=0;m<ecnt;m++){
    for(n=0;n<ecnt;n++){
      t=0;
      for(l=1;l<10;l++){if(teu[m][l]==teu[n][l]){t++;}}
      mtc[m][n]=t; scmtc[m][n]=t;
      t=0;
      for(l=1;l<10;l++){if(teu[m][l]+teu[n][l]==2){t++;}}
      cmtc[m][n]=t;
      if(scmtc[m][n]<t){scmtc[m][n]=t;}
    }
  }
  printf("%7d/%7d/%7d/%7d/\n",1,2,3,4);
  for(l=0;l<3;l++){l3=l*3;
    for(m=0;m<ecnt;m++){
      printf("  ");
      for(n=1;n<4;n++){printf("%2d",teu[m][l3+n]);}
    }
    printf("\n");
  }
  printf(" [Count of Euler Units = %d]\n",ecnt);
  printf("\n");
  printf(" [Reference Table for the Best Match]\n");
  printf("  S|");
  for(n=0;n<4;n++){printf("%3d",n+1);}
  printf("    C|");
  for(n=0;n<4;n++){printf("%3d",n+1);}
  printf("   SC|");
  for(n=0;n<4;n++){printf("%3d",n+1);}
  printf("\n");
  printf(" --+-------------  --+-------------  --+-------------\n");
  for(m=0;m<4;m++){
    printf("%3d|",m+1);
    for(n=0;n<4;n++){printf("%3d",mtc[m][n]);}
    printf("  %3d|",m+1);
    for(n=0;n<4;n++){printf("%3d",cmtc[m][n]);}
    printf("  %3d|",m+1);
    for(n=0;n<4;n++){printf("%3d",scmtc[m][n]);}
    printf("\n");
  }
}
//
//* Combine, Compose and Print *
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<ecnt;u1++){
    for(u2=0;u2<ecnt;u2++){
      if(scmtc[u1][u2]==3){
        for(n=1;n<10;n++){uflg[n]=0;}
        for(n=0;n<10;n++){
          d=teu[u1][n]*3+teu[u2][n]+1;
          nm[n]=d; uflg[d]++;
        }
        fc=0;
```

```
          for(n=1;n<10;n++){if(uflg[n]==0){fc++;}}
          if(fc==0){recprans();}
        }
      }
    }
}
//
//* Record and Print the Solution List *
void recprans(){
 short n;
  solt[cnt][0]=cnt+1; solt[cnt][10]=u1+1; solt[cnt][11]=u2+1;
  for(n=1;n<10;n++){solt[cnt][n]=nm[n];}
  cnt++;
  anm[0][0]=cnt;
  for(n=1;n<10;n++){tn[n]=nm[n]; anm[0][n]=tn[n];}; chksms(0);
  for(n=1;n<10;n++){tn[n]=nm[n]-1; anm[1][n]=tn[n];}; chksms(1);
  for(n=1;n<10;n++){tn[n]=teu[u1][n]; anm[2][n]=tn[n];}; chksms(2);
  for(n=1;n<10;n++){tn[n]=teu[u2][n]; anm[3][n]=tn[n];}; chksms(3);
  anm[2][0]=u1+1; anm[3][0]=u2+1;
  anspr();
}
//
void anspr(){
 short l,l3,n;
  printf(" /EU %dH\\%d/%d|     %dL\\%d/%d|  [Math]  \\%2d/%2d|%9d#\\%2d/%2d|",
          anm[2][0],cs[2][7],cs[2][8],anm[3][0],cs[3][7],cs[3][8],
          cs[1][7],cs[1][8],anm[0][0],cs[0][7],cs[0][8]);
  printf("\n");
  for(l=0;l<3;l++){l3=l*3;
    printf(" ");
    for(n=1;n<4;n++){printf("%2d",anm[2][l3+n]);}
    printf("|%d|%d| ",cs[2][l+1],cs[2][l+4]);
    for(n=1;n<4;n++){printf("%2d",anm[3][l3+n]);}
    printf("|%d|%d| ",cs[3][l+1],cs[3][l+4]);
    for(n=1;n<4;n++){printf("%3d",anm[1][l3+n]);}
    printf("|%2d|%2d| ",cs[1][l+1],cs[1][l+4]);
    for(n=1;n<4;n++){printf("%3d",anm[0][l3+n]);}
    printf("|%2d|%2d|",cs[0][l+1],cs[0][l+4]);
    printf("\n");
  }
}
//
//* Check Sums *
void chksms(short x){
  cs[x][1]=tn[1]+tn[2]+tn[3]; cs[x][2]=tn[4]+tn[5]+tn[6]; cs[x][3]=tn[7]+tn[8]+tn[9];
  cs[x][4]=tn[1]+tn[4]+tn[7]; cs[x][5]=tn[2]+tn[5]+tn[8]; cs[x][6]=tn[3]+tn[6]+tn[9];
  cs[x][7]=tn[1]+tn[5]+tn[9]; cs[x][8]=tn[3]+tn[5]+tn[7];
}
//
```

Let me show you the execution result of this program above.

```
** Reconstruct Standard Magic Squares of Order 3 by
    the 3rd Increment Number System and New Euler's Method **
 ** List of Euler Units **
      1/       2/       3/       4/
  0 2 1    1 0 2    1 2 0    2 0 1
  2 1 0    2 1 0    0 1 2    0 1 2
  1 0 2    0 2 1    2 0 1    1 2 0
  [Count of Euler Units = 4]
```

We have really got the Euler Units as many as 4. Then we can pick up any two of them to combine and compose the whole object by our arithmetic calculation.

Let me list out the primitive result at first, only built straight by 4x4 combinations without using any 'filters' to have only got the correct answers.

```
** Primitive List of Standard Magic Squares of Order 3 Reconstructed **
 [/Euler Units: High| Low|; [Math] Classic#;
      with Sum Checks of Each: \D1/D2|Row|Column|;]
 /EU 1H\3/3|     1L\3/3|  [Math]  \12/12|        1#\15/15|
  0 2 1|3|3|  0 2 1|3|3|   0  8  4|12|12|   1  9  5|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  1 0 2|3|3|  1 0 2|3|3|   4  0  8|12|12|   5  1  9|15|15|
 /EU 1H\3/3|     2L\3/3|  [Math]  \12/12|        2#\15/15|
  0 2 1|3|3|  1 0 2|3|3|   1  6  5|12|12|   2  7  6|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  1 0 2|3|3|  0 2 1|3|3|   3  2  7|12|12|   4  3  8|15|15|
 /EU 1H\3/3|     3L\3/3|  [Math]  \12/12|        3#\15/15|
  0 2 1|3|3|  1 2 0|3|3|   1  8  3|12|12|   2  9  4|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  1 0 2|3|3|  2 0 1|3|3|   5  0  7|12|12|   6  1  8|15|15|
 /EU 1H\3/3|     4L\3/3|  [Math]  \12/12|        4#\15/15|
  0 2 1|3|3|  2 0 1|3|3|   2  6  4|12|12|   3  7  5|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  1 0 2|3|3|  1 2 0|3|3|   4  2  6|12|12|   5  3  7|15|15|
 /EU 2H\3/3|     1L\3/3|  [Math]  \12/12|        5#\15/15|
  1 0 2|3|3|  0 2 1|3|3|   3  2  7|12|12|   4  3  8|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  0 2 1|3|3|  1 0 2|3|3|   1  6  5|12|12|   2  7  6|15|15|
 /EU 2H\3/3|     2L\3/3|  [Math]  \12/12|        6#\15/15|
  1 0 2|3|3|  1 0 2|3|3|   4  0  8|12|12|   5  1  9|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  0 2 1|3|3|  0 2 1|3|3|   0  8  4|12|12|   1  9  5|15|15|
 /EU 2H\3/3|     3L\3/3|  [Math]  \12/12|        7#\15/15|
  1 0 2|3|3|  1 2 0|3|3|   4  2  6|12|12|   5  3  7|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  0 2 1|3|3|  2 0 1|3|3|   2  6  4|12|12|   3  7  5|15|15|
 /EU 2H\3/3|     4L\3/3|  [Math]  \12/12|        8#\15/15|
  1 0 2|3|3|  2 0 1|3|3|   5  0  7|12|12|   6  1  8|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  0 2 1|3|3|  1 2 0|3|3|   1  8  3|12|12|   2  9  4|15|15|
 /EU 3H\3/3|     1L\3/3|  [Math]  \12/12|        9#\15/15|
  1 2 0|3|3|  0 2 1|3|3|   3  8  1|12|12|   4  9  2|15|15|
  0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
  2 0 1|3|3|  1 0 2|3|3|   7  0  5|12|12|   8  1  6|15|15|
 /EU 3H\3/3|     2L\3/3|  [Math]  \12/12|       10#\15/15|
  1 2 0|3|3|  1 0 2|3|3|   4  6  2|12|12|   5  7  3|15|15|
  0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
  2 0 1|3|3|  0 2 1|3|3|   6  2  4|12|12|   7  3  5|15|15|
 /EU 3H\3/3|     3L\3/3|  [Math]  \12/12|       11#\15/15|
  1 2 0|3|3|  1 2 0|3|3|   4  8  0|12|12|   5  9  1|15|15|
  0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
  2 0 1|3|3|  2 0 1|3|3|   8  0  4|12|12|   9  1  5|15|15|
```

```
/EU 3H\3/3|     4L\3/3|  [Math]  \12/12|       12#\15/15|
 1 2 0|3|3|  2 0 1|3|3|   5  6  1|12|12|   6  7  2|15|15|
 0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
 2 0 1|3|3|  1 2 0|3|3|   7  2  3|12|12|   8  3  4|15|15|
/EU 4H\3/3|     1L\3/3|  [Math]  \12/12|       13#\15/15|
 2 0 1|3|3|  0 2 1|3|3|   6  2  4|12|12|   7  3  5|15|15|
 0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
 1 2 0|3|3|  1 0 2|3|3|   4  6  2|12|12|   5  7  3|15|15|
/EU 4H\3/3|     2L\3/3|  [Math]  \12/12|       14#\15/15|
 2 0 1|3|3|  1 0 2|3|3|   7  0  5|12|12|   8  1  6|15|15|
 0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
 1 2 0|3|3|  0 2 1|3|3|   3  8  1|12|12|   4  9  2|15|15|
/EU 4H\3/3|     3L\3/3|  [Math]  \12/12|       15#\15/15|
 2 0 1|3|3|  1 2 0|3|3|   7  2  3|12|12|   8  3  4|15|15|
 0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
 1 2 0|3|3|  2 0 1|3|3|   5  6  1|12|12|   6  7  2|15|15|
/EU 4H\3/3|     4L\3/3|  [Math]  \12/12|       16#\15/15|
 2 0 1|3|3|  2 0 1|3|3|   8  0  4|12|12|   9  1  5|15|15|
 0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
 1 2 0|3|3|  1 2 0|3|3|   4  8  0|12|12|   5  9  1|15|15|
[Solution Counts = 16]  [OK!]
```

As you see, this list contains 8 incorrect answers, such as 1#, 4#, 6#, 7#, 10#, 11#, 13#, 16#. They are all against the basic rule that every number of {1, 2, 3, 4, 5, 6, 7, 8, 9} must be used. Neither duplication nor lack of any number is permitted.

Why do we have those incorrect answers? What combinations make them wrong?

Such the combinations of Euler Units as (1H, 1L), (2H, 2L), (3H, 3L), (4H, 4L) make the answers wrong. It means you must not take the same unit twice.

What do they mean by the combinations (1H, 4L), (2H, 3L), (3H, 2L), (4H, 1L)?

They mean the combinations of 'Complementary Units of 2'. If you pick up any number of one unit and add the corresponding number in the other, then they always add up to the constant sum 2: AH+BL=2, you should see. Complementary Units have those Complementary Numbers of 2. You must not take those pair units to combine.

How do we have to accept and express these new rules in our program, then?

I propose you to make such a reference table for the best combinations as follows:

```
[Reference Table for the Best Match]
 S|  1  2  3  4    C|  1  2  3  4   SC|  1  2  3  4
--+------------  --+------------  --+------------
 1|  9  3  3  3    1|  3  3  3  9    1|  9  3  3  9
 2|  3  9  3  3    2|  3  3  9  3    2|  3  9  9  3
 3|  3  3  9  3    3|  3  9  3  3    3|  3  9  9  3
 4|  3  3  3  9    4|  9  3  3  3    4|  9  3  3  9
```

We need such the last 'Collective Table of SC' as above for our reference, where we can only select the appropriate pair units, whose count are 3 in the table.

I dictated the necessary program at the top of the procedure void preunit(); and actually use it in the procedure void cmbcmp(); Study the last list of Program, please.

Here you see the list of all the correct reconstructions.

```
** List of Standard Magic Squares of Order 3 Reconstructed **
 [/Euler Units: High| Low|; [Math]  [Classic]; Sum Checks of Each: |D1\D2/Row|Column|;]
```

```
 /EU 1H\3/3|     2L\3/3|  [Math]  \12/12|        1#\15/15|
  0 2 1|3|3|  1 0 2|3|3|   1  6  5|12|12|   2  7  6|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  1 0 2|3|3|  0 2 1|3|3|   3  2  7|12|12|   4  3  8|15|15|
 /EU 1H\3/3|     3L\3/3|  [Math]  \12/12|        2#\15/15|
  0 2 1|3|3|  1 2 0|3|3|   1  8  3|12|12|   2  9  4|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  1 0 2|3|3|  2 0 1|3|3|   5  0  7|12|12|   6  1  8|15|15|
 /EU 2H\3/3|     1L\3/3|  [Math]  \12/12|        3#\15/15|
  1 0 2|3|3|  0 2 1|3|3|   3  2  7|12|12|   4  3  8|15|15|
  2 1 0|3|3|  2 1 0|3|3|   8  4  0|12|12|   9  5  1|15|15|
  0 2 1|3|3|  1 0 2|3|3|   1  6  5|12|12|   2  7  6|15|15|
 /EU 2H\3/3|     4L\3/3|  [Math]  \12/12|        4#\15/15|
  1 0 2|3|3|  2 0 1|3|3|   5  0  7|12|12|   6  1  8|15|15|
  2 1 0|3|3|  0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  0 2 1|3|3|  1 2 0|3|3|   1  8  3|12|12|   2  9  4|15|15|
 /EU 3H\3/3|     1L\3/3|  [Math]  \12/12|        5#\15/15|
  1 2 0|3|3|  0 2 1|3|3|   3  8  1|12|12|   4  9  2|15|15|
  0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
  2 0 1|3|3|  1 0 2|3|3|   7  0  5|12|12|   8  1  6|15|15|
 /EU 3H\3/3|     4L\3/3|  [Math]  \12/12|        6#\15/15|
  1 2 0|3|3|  2 0 1|3|3|   5  6  1|12|12|   6  7  2|15|15|
  0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
  2 0 1|3|3|  1 2 0|3|3|   7  2  3|12|12|   8  3  4|15|15|
 /EU 4H\3/3|     2L\3/3|  [Math]  \12/12|        7#\15/15|
  2 0 1|3|3|  1 0 2|3|3|   7  0  5|12|12|   8  1  6|15|15|
  0 1 2|3|3|  2 1 0|3|3|   2  4  6|12|12|   3  5  7|15|15|
  1 2 0|3|3|  0 2 1|3|3|   3  8  1|12|12|   4  9  2|15|15|
 /EU 4H\3/3|     3L\3/3|  [Math]  \12/12|        8#\15/15|
  2 0 1|3|3|  1 2 0|3|3|   7  2  3|12|12|   8  3  4|15|15|
  0 1 2|3|3|  0 1 2|3|3|   0  4  8|12|12|   1  5  9|15|15|
  1 2 0|3|3|  2 0 1|3|3|   5  6  1|12|12|   6  7  2|15|15|
 [Solution Counts = 8]   [OK!]
** Calculated and Listed by Kanji Setsuda
    on Apr.14, 2012 with MacOSX 10.7.3 & Xcode 4.3.2 **
```

These 8 reconstructions are no longer wrong ones, but they are considered to be still the ‘primitive solutions’. Each of them is such an imitation of any other one as made by simple reflection or rotation. It is not the set of independent solutions.

You must use some filter program to select only the ‘independent solution’ out of those primitive eight, such as the next ‘list-forming inequality conditions complex’.

```
fc=0;
for(n=1;n<10;n++){if(uflg[n]==0){fc++;}}
if(fc==0){recprans();}
```

Find this part in the procedure void cmbcmp(); and rewrite it as below, for instance.

```
fc=0;
for(n=1;n<10;n++){if(uflg[n]==0){fc++;}}
if(fc==0){
  if((nm[1]<nm[3]&&(nm[3]<nm[7]&&(nm[1]<nm[9]){recprans();}
```

```
** List of the only Standard Solution of Order 3 Reconstructed **
 [/Euler Units: High| Low|; [Math]  [Classic]; Sum Checks of Each: |D1\D2/Row|Column|;]
 /EU 1H\3/3|      3L\3/3|  [Math]  \12/12|        1#\15/15|
  0 2 1|3|3|   1 2 0|3|3|   1  8  3|12|12|   2  9  4|15|15|
  2 1 0|3|3|   0 1 2|3|3|   6  4  2|12|12|   7  5  3|15|15|
  1 0 2|3|3|   2 0 1|3|3|   5  0  7|12|12|   6  1  8|15|15|
 [Solution Counts = 1]   [OK!]
```

We have got it! It is certain that there is only one answer for this type, as you know.

I am afraid you might feel too hard to follow this big experiment, just for the only magic square 3x3. You might think it is a little too ridiculous. Of course, you don't have to do the same thing again by yourself. You may well only read this report of 'academic study' and understand that our New Euler's Method is effective here.

Every step demonstrates quite reasonable, and certifies the reconstruction result is just the same with the one we got before in our old study.

We have just come to know the Standard Magic Square 3x3 has the beautiful 'inner structure'. Though it fails to be the 'Complete Euler Square', it proves to be the beautiful '**Euler Square**' of Order 3 successfully. We may call that, don't you think?

## 2. How about Magic Cube of Order 3?

Let's build Magic Cubes of Order 3 by this New Euler's Method using the Positional Number System of Base 3, namely the 3rd Increment Number System.

Let me show you the conceptual diagrams of our method at first.

```
** Conceptual Diagrams for the New Euler's Method **
 [Decomposition of Self-Complementary Magic Cubes of Order 3]
 ( Classical#, [Mathematical] Decomposed Units:/D3i: High/ Middle/ Low/;)
  *#                  [Math]              /D3i    H/          M/          L/
   1-----15-----26     0-----14-----25     0---1---2   0---1---2   0---2---1
   |23     7    |12    |22     6    |11    |2   0  |1  |1   2  |0  |1   0  |2
   |  18-----20------4  |  17-----19------3  | 1---2---0 | 2---0---1 | 2---1---0
  17   | 19     6   |  16   | 18     5   |  1 | 2   0 | 2 | 0   1 | 1 | 0   2 |
   | 3 |   14    |25 |   | 2 |   13    |24 |  |0|  1  |2|  |0|  1  |2|  |2|  1  |0|
   |  22     9  |  11   |  21     8  |  10   |  2   0 | 1 |  1   2 | 0 |  0   2 | 1
  24---|--8-----10   |  23---|--7----- 9   |  2-|-0---1 |  1-|-2---0 |  2-|-1---0 |
    16 |   21     5 |    15 |   20     4 |    1|  2   0|   2|  0   1|   0|  2   1|
       2-----13-----27      1-----12-----26    0---1---2    0---1---2    1---0---2

 Calculation: Classic# - 1 = [Math];        3) 21  The Remainders put into /D3i
   Serial Division of [Math] is needed:     3)  7 ... 0 -> Low/
                                            3)  2 ... 1 -> Middle/
                                                0 ... 2 -> High/

 [Composition of Self-Complementary Magic Cubes of Order 3]
 (Euler Units: High/Middle/Low/; Reconstruction: [Math] Solution Number#)
 /EU    */H        **/M       ***/L   [Math]                          *#
  0---1---2   0---1---2   0---2---1    0-----14-----25      1-----15-----26
  |2   0  |1  |1   2  |0  |1   0  |2   |22     6    |11    |23     7    |12
  | 1---2---0 | 2---0---1 | 2---1---0  |  17-----19------3  |  18-----20------4
  1 | 2   0 | 2 | 0   1 | 1 | 0   2 | 16   | 18     5   | 17   | 19     6   |
  |0|  1  |2| |0|  1  |2| |2|  1  |0|  | 2 |   13    |24 |  | 3 |   14    |25 |
  | 2   0 | 1 | 1   2 | 0 | 0   2 | 1  |  21     8  |  10  |  22     9  |  11
  2-|-0---1 |  1-|-2---0 |  2-|-1---0 |  23---|--7----- 9   |  24---|--8-----10   |
   1|  2   0|   2|  0   1|   0|  2   1|    15 |   20     4 |   16 |   21     5 |
    0---1---2    0---1---2    1---0---2      1-----12-----26      2-----13-----27

 Calculation: (a/H x 3 + b/M) x 3 + c/L = [Math]; [Math] + 1 = Classic#;
   (0 x 3 + 1) x 3 + 2 = 5;       (2 x 3 + 1) x 3 + 0 = 21;     ...
```

We have to deal with such the three dimensional object now that we always have to make three units for both Decomposed ones and for Euler types in each case.

The necessary calculations must be more complex. You need the 'serial division' by 3, putting every successive remainder into decomposed units in the reverse order.

In the low process you must have the 'serial multiplications' by 3 as shown above.

First of all let's design and compose each Euler Unit for our object.

I prepared the following Basic Form and Basic Conditions for our definitions. I made after the original definitions of the whole object SCMC333 modifying only a little.

```
//
//** Basic Form and Definitions: C=3; **
//   1-----10-----19         n1+n2+n3=C;     | n1+n10+n19=C; |   n1+n4+n7=C;
//   | 2     11    |20       n4+n5+n6=C;     | n2+n11+n20=C; |   n2+n5+n8=C;
//   |   3-----12-----21     n7+n8+n9=C;     | n3+n12+n21=C; |   n3+n6+n9=C;
//   4   | 13     22   |     n10+n11+n12=C;   | n4+n13+n22=C; | n10+n13+n16=C;
//   | 5 |   14    |23 |     n13+n14+n15=C;   | n5+n14+n23=C; | n11+n14+n17=C;
//   |   6     15  |  24     n16+n17+n18=C;   | n6+n15+n24=C; | n12+n15+n18=C;
//   7---|-16-----25   |     n19+n20+n21=C;   | n7+n16+n25=C; | n19+n22+n25=C;
//     8 |   17     26 |     n22+n23+n24=C;   | n8+n17+n26=C; | n20+n23+n26=C;
//       9-----18-----27     n25+n26+n27=C;   | n9+n18+n27=C; | n21+n24+n27=C;
//** Self-complementary Conditions: SC=2;
//   n1+n27=n2+n26=n3+n25=n4+n24+n5+n23
//   =n6+n22=n7+n21=n8+n20=n9+n19=n10+n18
//   =n11+n17=n12+n16=n13+n15=n14+n14=SC ...scc
//
```

Our New Euler's Method needs this 'similar structure' between the whole object and its Euler Units. It is the fundamental presupposition of our **Euler Type** of objects.

In each unit the value of magic constants should be designed to be C=3 and SC=2.

I do not define anything now about the 4 primary triagonals:

```
n1+n14+n27 ... pt1;      n3+n14+n25 ... pt2;
n7+n14+n21 ... pt3;      n9+n14+n19 ... pt4;
```

Because every sum of these four triagonals should be surely equal to C, since n14=SC/2=1; and (n1+n27)+n14 = SC+1 = 2+1=3 = C; ... pt1; (n3+n25)+n14 = SC+1 = C; ... pt2; ... only according to the Self-Complementary Conditions above.

We do not define any content pattern either, about the constant sums of rows, columns and four triagonals. We do not have to always follow the Latin Conditions.

It may be {1, 1, 1}, or may be {0, 1, 2}. We don't care about that at all now.

But, we should take one more important rule and add it to the definition list above:

> "In each Unit any number of {0, 1, 2} must be used just **nine times** as often as the others. More or less often it must not be used."

Let me show you the program recently dictated and its execution results as follows.

```
//** Self-Complementary Magic Cubes of Order 3: Reconstructed  **
//** by the 3rd Increment N. System and the New Euler's Method **
//** 'NEulerMC3.c': in 2003 and 2005; Revised on Apr.17, 2012; **
//**  built by Kanji Setsuda with MacOSX 10.7.3 & Xcode 4.3.2  **
//
#include <stdio.h>
//
//* Variables *
short int cnt;
short SC, LSM;
short u1, u2, u3, u4;
short teu[9][28];
short mtc[9][9];
short nm[28], uflg[28];
//
```

```
//* Sub-Routines *
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);
void recordans(void);
void preunit(void), cmbcmp(void);
void prfmcans(void);
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("** Self-Complementary Magic Cubes of Order 3: Reconstructed by\n");
  printf("    the 3rd Increment Number System and the New Euler's Method **\n");
  for(n=0;n<3;n++){uflg[n]=0;}
  SC=2; LSM=3; cnt=0;
  nm[14]=1; uflg[1]=1;
  printf("\n");
  printf(" [List of Euler Units]\n");
  stp01();                    //* Making Euler Units *
  preunit();                  //* Print the Euler Units *
  printf("\n");
  printf(" [Reconstruction of Self-Complementary Magic Cubes of Order 3]\n");
  printf(" (Euler Units: High/Middle/Low/; Reconstruction: [Math] Sol Number#)\n");
  cnt=0;
  cmbcmp();                   //* Combine, Compose and Print *
  printf(" [Count of Solutions = %d]  [OK!]\n",cnt);
  printf(" ** Calculated and Listed by Kanji Setsuda\n");
  printf("     on Apr.17, 2012 with MacOSX 10.7.3 and Xcode 4.3.2\n");
  printf("\n");
 return 0;
}
//
//* Making Euler Units *
//Level #1:
//Set n1 & n27
void stp01(){
 short i,j;
  for(i=0;i<3;i++){j=SC-i;
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[1]=i; nm[27]=j;
      uflg[i]++; uflg[j]++;
      stp02();
      uflg[j]--; uflg[i]--;}
  }
}
//Set n2 & n26
void stp02(){
 short i,j;
  for(i=2;i>=0;i--){j=SC-i;
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[2]=i; nm[26]=j;
      uflg[i]++; uflg[j]++;
      stp03();
      uflg[j]--; uflg[i]--;}
  }
}
//Set n3=LSM-n1-n2 & n25
void stp03(){
 short i,j;
  i=LSM-nm[1]-nm[2];
  j=LSM-nm[27]-nm[26];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[3]=i; nm[25]=j;
      uflg[i]++; uflg[j]++;
      stp04();
      uflg[j]--; uflg[i]--;}}
}
//Set n4 & n24
void stp04(){
 short i,j;
  for(i=2;i>=0;i--){j=SC-i;
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[4]=i; nm[24]=j;
      uflg[i]++; uflg[j]++;
      stp05();
      uflg[j]--; uflg[i]--;}
  }
}
//Set n7=LSM-n1-n4 & n21
void stp05(){
 short i,j;
  i=LSM-nm[1]-nm[4];
  j=LSM-nm[27]-nm[24];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[7]=i; nm[21]=j;
      uflg[i]++; uflg[j]++;
      stp06();
      uflg[j]--; uflg[i]--;}}
}
//Set n10 & n18
void stp06(){
```

```
 short i,j;
  for(i=2;i>=0;i--){j=SC-i;
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[10]=i; nm[18]=j;
      uflg[i]++; uflg[j]++;
      stp07();
      uflg[j]--; uflg[i]--;}
  }
}
//Set n19=LSM-n1-n10 & n9
void stp07(){
 short i,j;
  i=LSM-nm[1]-nm[10];
  j=LSM-nm[27]-nm[18];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[19]=i; nm[9]=j;
      uflg[i]++; uflg[j]++;
      stp08();
      uflg[j]--; uflg[i]--;}}
}
//Level #2:
//Set n6=LSM-n3-n9 & n22
void stp08(){
 short i,j;
  i=LSM-nm[3]-nm[9];
  j=LSM-nm[25]-nm[19];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[6]=i; nm[22]=j;
      uflg[i]++; uflg[j]++;
      stp09();
      uflg[j]--; uflg[i]--;}}
}
//Set n5=LSM-n4-n6 & n23
void stp09(){
 short i,j;
  i=LSM-nm[4]-nm[6];
  j=LSM-nm[24]-nm[22];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[5]=i; nm[23]=j;
      uflg[i]++; uflg[j]++;
      stp10();
      uflg[j]--; uflg[i]--;}}
}
//Set n8=LSM-n2-n5 & n20
void stp10(){
 short i,j,k,l;
  i=LSM-nm[2]-nm[5];
  j=LSM-nm[26]-nm[23];
  k=LSM-nm[7]-nm[9];
  l=LSM-nm[21]-nm[19];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((i==k)&&(j==l)){
      if((uflg[i]<9)&&(uflg[j]<9)){
        nm[8]=i; nm[20]=j;
        uflg[i]++; uflg[j]++;
        stp11();
        uflg[j]--; uflg[i]--;}}}
}
//Level #3:
//Set n11=LSM-n2-n20 & n17
void stp11(){
 short i,j;
  i=LSM-nm[2]-nm[20];
  j=LSM-nm[26]-nm[8];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((uflg[i]<9)&&(uflg[j]<9)){
      nm[11]=i; nm[17]=j;
      uflg[i]++; uflg[j]++;
      stp12();
      uflg[j]--; uflg[i]--;}}
}
//Set n12=LSM-n10-n11 & n16
void stp12(){
 short i,j,k,l;
  i=LSM-nm[10]-nm[11];
  j=LSM-nm[18]-nm[17];
  k=LSM-nm[3]-nm[21];
  l=LSM-nm[25]-nm[7];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((i==k)&&(j==l)){
      if((uflg[i]<9)&&(uflg[j]<9)){
        nm[12]=i; nm[16]=j;
        uflg[i]++; uflg[j]++;
        stp13();
        uflg[j]--; uflg[i]--;}}}
}
//Set n13=LSM-n10-n16 & n15
void stp13(){
 short i,j,k,l;
  i=LSM-nm[10]-nm[16];
  j=LSM-nm[18]-nm[12];
  k=LSM-nm[4]-nm[22];
  l=LSM-nm[24]-nm[6];
  if((0<=i)&&(i<3)&&(i+j==SC)){
    if((i==k)&&(j==l)){
      if((uflg[i]<9)&&(uflg[j]<9)){
        nm[13]=i; nm[15]=j;
        uflg[i]++; uflg[j]++;
        stp14();
        uflg[j]--; uflg[i]--;}}}
}
```

---

```
//
//* The Last Checks *
void stp14(){
 short sm1,sm2,sm3;
  sm1=nm[5]+nm[14]+nm[23];
  sm2=nm[11]+nm[14]+nm[17];
  sm3=nm[13]+nm[14]+nm[15];
  if((sm1==LSM)&&(sm2==LSM)&&(sm2==LSM)){recordans();}
}
//
//* Record the Euler Units */
void recordans(){
```

```
 short n;
  teu[cnt][0]=cnt+1;
  for(n=1;n<28;n++){teu[cnt][n]=nm[n];}
  cnt++;
}
//
//* Print the Euler Units *
void preunit(){
 short l,m,n,t;
  for(m=0;m<cnt;m=m+4){
    printf("%10d/%12d/%12d/%12d/\n",
            teu[m][0],teu[m+1][0],teu[m+2][0],teu[m+3][0]);
    printf("%3d---%d---%d",teu[m][1],teu[m][10],teu[m][19]);
    printf("%5d---%d---%d",teu[m+1][1],teu[m+1][10],teu[m+1][19]);
    printf("%5d---%d---%d",teu[m+2][1],teu[m+2][10],teu[m+2][19]);
    printf("%5d---%d---%d\n",teu[m+3][1],teu[m+3][10],teu[m+3][19]);
    printf("  |%d%4d  |%d",teu[m][2],teu[m][11],teu[m][20]);
    printf("   |%d%4d  |%d",teu[m+1][2],teu[m+1][11],teu[m+1][20]);
    printf("   |%d%4d  |%d",teu[m+2][2],teu[m+2][11],teu[m+2][20]);
    printf("   |%d%4d  |%d\n",teu[m+3][2],teu[m+3][11],teu[m+3][20]);
    printf("  |%2d---%d---%d",teu[m][3],teu[m][12],teu[m][21]);
    printf("  |%2d---%d---%d",teu[m+1][3],teu[m+1][12],teu[m+1][21]);
    printf("  |%2d---%d---%d",teu[m+2][3],teu[m+2][12],teu[m+2][21]);
    printf("  |%2d---%d---%d\n",teu[m+3][3],teu[m+3][12],teu[m+3][21]);
    printf("%3d |%2d%4d |",teu[m][4],teu[m][13],teu[m][22]);
    printf("%3d |%2d%4d |",teu[m+1][4],teu[m+1][13],teu[m+1][22]);
    printf("%3d |%2d%4d |",teu[m+2][4],teu[m+2][13],teu[m+2][22]);
    printf("%3d |%2d%4d |\n",teu[m+3][4],teu[m+3][13],teu[m+3][22]);
    printf("  |%d|%3d  |%d|",teu[m][5],teu[m][14],teu[m][23]);
    printf("  |%d|%3d  |%d|",teu[m+1][5],teu[m+1][14],teu[m+1][23]);
    printf("  |%d|%3d  |%d|",teu[m+2][5],teu[m+2][14],teu[m+2][23]);
    printf("  |%d|%3d  |%d|\n",teu[m+3][5],teu[m+3][14],teu[m+3][23]);
    printf("  |%2d%4d |%2d",teu[m][6],teu[m][15],teu[m][24]);
    printf("  |%2d%4d |%2d",teu[m+1][6],teu[m+1][15],teu[m+1][24]);
    printf("  |%2d%4d |%2d",teu[m+2][6],teu[m+2][15],teu[m+2][24]);
    printf("  |%2d%4d |%2d\n",teu[m+3][6],teu[m+3][15],teu[m+3][24]);
    printf("%3d-|-%d---%d |",teu[m][7],teu[m][16],teu[m][25]);
    printf("%3d-|-%d---%d |",teu[m+1][7],teu[m+1][16],teu[m+1][25]);
    printf("%3d-|-%d---%d |",teu[m+2][7],teu[m+2][16],teu[m+2][25]);
    printf("%3d-|-%d---%d |\n",teu[m+3][7],teu[m+3][16],teu[m+3][25]);
    printf("%4d|%3d%4d|",teu[m][8],teu[m][17],teu[m][26]);
    printf("%4d|%3d%4d|",teu[m+1][8],teu[m+1][17],teu[m+1][26]);
    printf("%4d|%3d%4d|",teu[m+2][8],teu[m+2][17],teu[m+2][26]);
    printf("%4d|%3d%4d|\n",teu[m+3][8],teu[m+3][17],teu[m+3][26]);
    printf("%5d---%d---%d",teu[m][9],teu[m][18],teu[m][27]);
    printf("%5d---%d---%d",teu[m+1][9],teu[m+1][18],teu[m+1][27]);
    printf("%5d---%d---%d",teu[m+2][9],teu[m+2][18],teu[m+2][27]);
    printf("%5d---%d---%d\n",teu[m+3][9],teu[m+3][18],teu[m+3][27]);
    printf("\n");
  }
  printf(" [Count of Euler Units = %d]\n",cnt);
 //* Check each pair how well they match *
  for(m=0;m<cnt;m++){
    for(l=0;l<cnt;l++){
      t=0;
      for(n=1;n<28;n++){if(teu[m][n]==teu[l][n]){t++;}}
      mtc[m][l]=t;
      t=0;
      for(n=1;n<28;n++){if(teu[m][n]+teu[l][n]==SC){t++;}}
      if(mtc[m][l]<t){mtc[m][l]=t;}
    }}
 //* Print the Reference Table *
  printf("\n");
  printf(" [Reference Table for the Best Matching]\n");
  printf(" SC|");
```

```
   for(n=1;n<=cnt;n++){printf("%3d",n);}
   printf("\n");
   printf(" --+------------------------\n");
   for(m=0;m<cnt;m++){
     printf("%3d|",m+1);
     for(n=0;n<cnt;n++){printf("%3d",mtc[m][n]);}
     printf("\n");
   }
}
//
//* Combine and Compose *
void cmbcmp(){
 short n,d,lu,md,fc;
   lu=8; md=9;
   for(u1=0;u1<lu;u1++){
     for(u2=0;u2<lu;u2++){
       if(mtc[u2][u1]==md){
         for(u3=0;u3<lu;u3++){
           if((mtc[u3][u1]==md)&&(mtc[u3][u2]==md)){
             for(n=1;n<28;n++){uflg[n]=0;}
             for(n=1;n<28;n++){
               d=teu[u1][n]*9+teu[u2][n]*3+teu[u3][n]+1;
               nm[n]=d; uflg[d]++;
             }
             fc=0;
             for(n=1;n<28;n++){
               if(uflg[n]==1){fc++;}else{break;}
             }
             if(fc==27){
               if((nm[1]<nm[3])&&(nm[1]<nm[7])&&(nm[1]<nm[9])&&(nm[1]<nm[19])){
                 if((nm[1]<nm[21])&&(nm[1]<nm[25])&&(nm[1]<nm[27])){
                   if((nm[2]>nm[4])&&(nm[4]>nm[10])){prfmcans();
                   }}}}
         }
       }
     }
   }
}
//
//* Print every Forms of the Answer *
void prfmcans(){
   cnt++;
   printf("%10d/%12d/%12d/    [Math]%30d#\n",u1+1,u2+1,u3+1,cnt);
   printf("%3d---%d---%d",teu[u1][1],teu[u1][10],teu[u1][19]);
   printf("%5d---%d---%d",teu[u2][1],teu[u2][10],teu[u2][19]);
   printf("%5d---%d---%d",teu[u3][1],teu[u3][10],teu[u3][19]);
   printf("%6d-----%2d-----%2d",nm[1]-1,nm[10]-1,nm[19]-1);
   printf("%7d-----%2d-----%2d\n",nm[1],nm[10],nm[19]);
   printf("  |%d%4d  |%d",teu[u1][2],teu[u1][11],teu[u1][20]);
   printf("   |%d%4d  |%d",teu[u2][2],teu[u2][11],teu[u2][20]);
   printf("   |%d%4d  |%d",teu[u3][2],teu[u3][11],teu[u3][20]);
   printf("    |%2d%7d    |%2d",nm[2]-1,nm[11]-1,nm[20]-1);
   printf("    |%2d%7d    |%2d\n",nm[2],nm[11],nm[20]);
   printf("  |%2d---%d---%d",teu[u1][3],teu[u1][12],teu[u1][21]);
   printf("  |%2d---%d---%d",teu[u2][3],teu[u2][12],teu[u2][21]);
   printf("  |%2d---%d---%d",teu[u3][3],teu[u3][12],teu[u3][21]);
   printf("   |%4d-----%2d------%d",nm[3]-1,nm[12]-1,nm[21]-1);
   printf("  |%4d-----%2d------%d\n",nm[3],nm[12],nm[21]);
   printf("%3d |%2d%4d",teu[u1][4],teu[u1][13],teu[u1][22]);
   printf(" |%3d |%2d%4d",teu[u2][4],teu[u2][13],teu[u2][22]);
   printf(" |%3d |%2d%4d",teu[u3][4],teu[u3][13],teu[u3][22]);
   printf(" |%4d   | %2d%7d   |",nm[4]-1,nm[13]-1,nm[22]-1);
   printf("%3d   | %2d%7d   |\n",nm[4],nm[13],nm[22]);
   printf("  |%d|%3d  |%d|",teu[u1][5],teu[u1][14],teu[u1][23]);
```

```
  printf("  |%d|%3d  |%d|",teu[u2][5],teu[u2][14],teu[u2][23]);
  printf("  |%d|%3d  |%d|",teu[u3][5],teu[u3][14],teu[u3][23]);
  printf("   |%2d |%5d    |%2d |",nm[5]-1,nm[14]-1,nm[23]-1);
  printf("  |%2d |%5d    |%2d |\n",nm[5],nm[14],nm[23]);
  printf("  |%2d%4d |%2d",teu[u1][6],teu[u1][15],teu[u1][24]);
  printf("  |%2d%4d |%2d",teu[u2][6],teu[u2][15],teu[u2][24]);
  printf("  |%2d%4d |%2d",teu[u3][6],teu[u3][15],teu[u3][24]);
  printf("   |%4d%7d  |%4d",nm[6]-1,nm[15]-1,nm[24]-1);
  printf("  |%4d%7d  |%4d\n",nm[6],nm[15],nm[24]);
  printf("%3d-|-%d---%d |",teu[u1][7],teu[u1][16],teu[u1][25]);
  printf("%3d-|-%d---%d |",teu[u2][7],teu[u2][16],teu[u2][25]);
  printf("%3d-|-%d---%d |",teu[u3][7],teu[u3][16],teu[u3][25]);
  printf("%4d---|--%d-----%2d   |",nm[7]-1,nm[16]-1,nm[25]-1);
  printf("%3d---|--%d-----%2d   |\n",nm[7],nm[16],nm[25]);
  printf("%4d|%3d%4d|",teu[u1][8],teu[u1][17],teu[u1][26]);
  printf("%4d|%3d%4d|",teu[u2][8],teu[u2][17],teu[u2][26]);
  printf("%4d|%3d%4d|",teu[u3][8],teu[u3][17],teu[u3][26]);
  printf("%6d |%5d%7d |",nm[8]-1,nm[17]-1,nm[26]-1);
  printf("%5d |%5d%7d |\n",nm[8],nm[17],nm[26]);
  printf("%5d---%d---%d",teu[u1][9],teu[u1][18],teu[u1][27]);
  printf("%5d---%d---%d",teu[u2][9],teu[u2][18],teu[u2][27]);
  printf("%5d---%d---%d",teu[u3][9],teu[u3][18],teu[u3][27]);
  printf("%8d-----%2d-----%2d",nm[9]-1,nm[18]-1,nm[27]-1);
  printf("%7d-----%2d-----%2d\n",nm[9],nm[18],nm[27]);
  printf("\n");
}
//
```

** Self-Complementary Magic Cubes of Order 3: Reconstructed by the 3rd Increment Number System and the New Euler's Method **

[List of Euler Units]

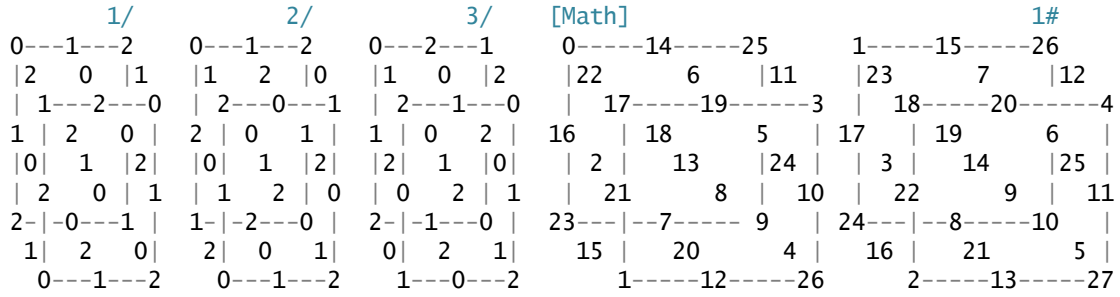
```
        1/          2/          3/          4/
 0---1---2   0---1---2   0---2---1   1---2---0
 |2   0  |1  |1   2  |0  |1   0  |2  |2   0  |1
 | 1---2---0 | 2---0---1 | 2---1---0 | 0---1---2
1 | 2   0 | 2 | 0   1 | 1 | 0   2 | 2 | 0   1 |
|0|  1  |2| |0|  1  |2| |2|  1  |0| |0|  1  |2|
| 2   0 | 1 | 1   2 | 0 | 0   2 | 1 | 1   2 | 0
2-|-0---1 | 1-|-2---0 | 2-|-1---0 | 0-|-1---2 |
 1|  2   0|  2|  0   1|  0|  2   1|  1|  2   0|
  0---1---2   0---1---2   1---0---2   2---0---1

        5/          6/          7/          8/
 1---0---2   2---0---1   2---1---0   2---1---0
 |0   2  |1  |1   2  |0  |1   0  |2  |0   2  |1
 | 2---1---0 | 0---1---2 | 0---2---1 | 1---0---2
0 | 2   1 | 1 | 2   0 | 0 | 2   1 | 1 | 0   2 |
|2|  1  |0| |0|  1  |2| |2|  1  |0| |2|  1  |0|
| 1   0 | 2 | 2   0 | 1 | 1   0 | 2 | 0   2 | 1
2-|-1---0 | 0-|-1---2 | 1-|-0---2 | 0-|-2---1 |
 1|  0   2|  2|  0   1|  0|  2   1|  1|  0   2|
  0---2---1   1---2---0   2---1---0   2---1---0
```

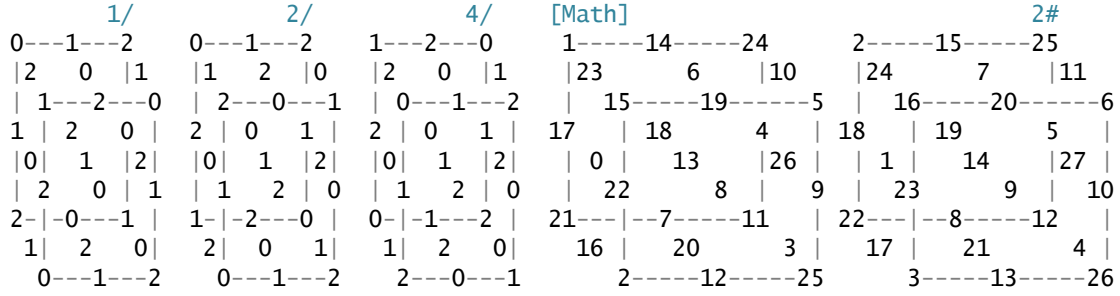
[Count of Euler Units = 8]

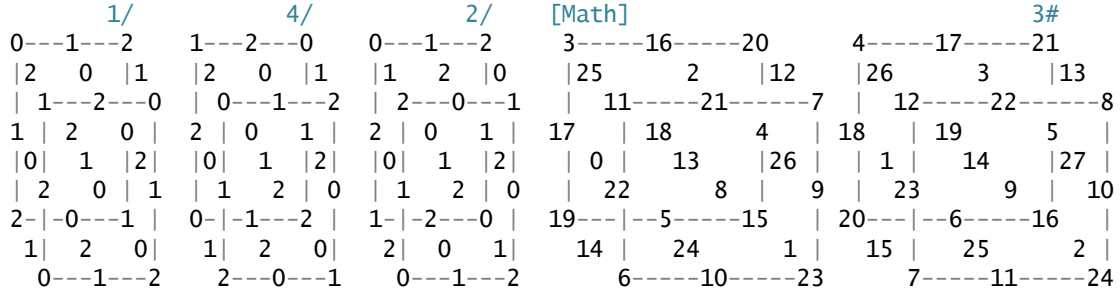
[Reference Table for the Best Matching]

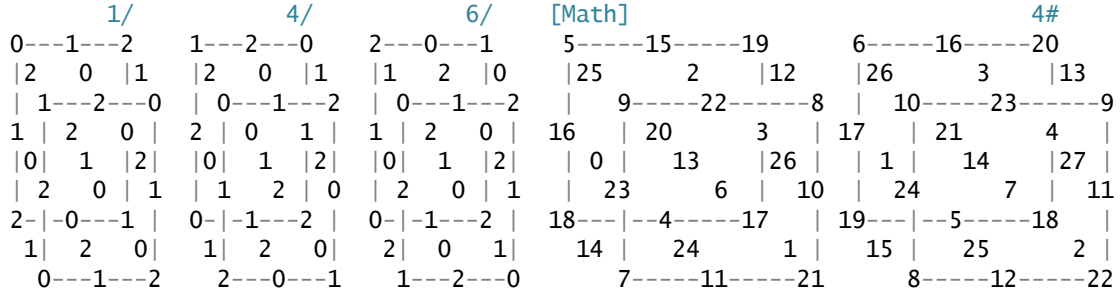
| SC | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 1 | 27 | 9 | 9 | 9 | 9 | 9 | 9 | 27 |
| 2 | 9 | 27 | 9 | 9 | 9 | 9 | 27 | 9 |
| 3 | 9 | 9 | 27 | 9 | 9 | 27 | 9 | 9 |
| 4 | 9 | 9 | 9 | 27 | 27 | 9 | 9 | 9 |
| 5 | 9 | 9 | 9 | 27 | 27 | 9 | 9 | 9 |
| 6 | 9 | 9 | 27 | 9 | 9 | 27 | 9 | 9 |
| 7 | 9 | 27 | 9 | 9 | 9 | 9 | 27 | 9 |
| 8 | 27 | 9 | 9 | 9 | 9 | 9 | 9 | 27 |

```
[Reconstruction of Self-Complementary Magic Cubes of Order 3]
(Euler Units: High/Middle/Low/; Reconstruction: [Math] Solution Number#)
```




```
[Count of Solutions = 4]  [OK!]
** Calculated and Listed by Kanji Setsuda
    on Apr.17, 2012 with MacOSX 10.7.3 and Xcode 4.3.2
```

All these reconstructions prove to be just the same with the ones we got before.

Our New Euler's Method could successfully compose them as the beautiful "**Euler Cubes**" and could surely demonstrate how effective it is to clarify the real 'inner structure' of our whole objects.

## 3. How to Make Pan-Diagonal MS55 by the 5th Increment Number System

Let's examine our New Euler's Method applying to the case of Pan-Diagonal Magic Squares of Order 5 and also the Simultaneous type next in this section.

We once built them by our 'Knight's Tour Composition', but we have not yet applied our New Euler's Method to these objects.

Let's build them for the "**Complete Euler Squares**" of Order 5. We already know that every row, every column and every pan-diagonal in each Euler Unit should have

the only content pattern (0, 1, 2, 3, 4), and should consequently add up to the same constant sum: 0+1+2+3+4=10. We can make nothing but this type.

```
** Conceptual Diagrams for the New Euler's Method **
 [Decomposition by the 5th Increment Number System]
 [Classic]   *#   [Math]        |Rw|Cl\P1/P2| /D5i    *H|Rw|Cl\P1/P2|        **L|Rw|Cl\P1/P2|
  1 23 20 12  9    0 22 19 11  8|60|60|60|60|  0 4 3 2 1|10|10|10|10|  0 2 4 1 3|10|10|10|10|
 15  7  4 21 18   14  6  3 20 17|60|60|60|60|  2 1 0 4 3|10|10|10|10|  4 1 3 0 2|10|10|10|10|
 24 16 13 10  2   23 15 12  9  1|60|60|60|60|  4 3 2 1 0|10|10|10|10|  3 0 2 4 1|10|10|10|10|
  8  5 22 19 11    7  4 21 18 10|60|60|60|60|  1 0 4 3 2|10|10|10|10|  2 4 1 3 0|10|10|10|10|
 17 14  6  3 25   16 13  5  2 24|60|60|60|60|  3 2 1 0 4|10|10|10|10|  1 3 0 2 4|10|10|10|10|
(Calculation: Divide [Math] by 5, and put the integer part of quotient into /D5i High unit
              and put the remainder into Low.)

 [Composition by the 5th Increment Number System]
 /LU   */H        **/L   [Math]        |Rw|Cl\P1/P2|  [Classic]   *#|Rw|Cl\P1/P2|
 0 4 3 2 1   0 2 4 3 1   0 22 19 13  6|60|60|60|60|   1 23 20 14  7|65|65|65|65|
 2 1 0 4 3   4 3 1 0 2  14  8  1 20 17|60|60|60|60|  15  9  2 21 18|65|65|65|65|
 4 3 2 1 0   1 0 2 4 3  21 15 12  9  3|60|60|60|60|  22 16 13 10  4|65|65|65|65|
 1 0 4 3 2   2 4 3 1 0   7  4 23 16 10|60|60|60|60|   8  5 24 17 11|65|65|65|65|
 3 2 1 0 4   3 1 0 2 4  18 11  5  2 24|60|60|60|60|  19 12  6  3 25|65|65|65|65|
(Calculation: m/H x 5 + n/L = [Math]; [Math] + 1 = [Classic] in the corresponding positions:
              4 x 5 + 2 = 22; 22 + 1 = 23;    2 x 5 + 4 = 14; 14 + 1 = 15; ...
```

Let's build each '**Latin Unit**' under that condition at first, and then pick up any two of them to combine and compose our objective solutions in the end.

What definitions do we have to give to those Latin Units, then?

I prepared such the basic form and basic conditions in the same style as before.

```
//
//** Basic Form for the Pan-diagonal Magic Squares of Order 5 **
//            .--.--.--.--.--.
//  2  3  4  5| 1| 2| 3| 4| 5| 1  2  3  4
//            |--+--+--+--+--|
//  7  8  9 10| 6| 7| 8| 9|10| 6  7  8  9
//            |--+--+--+--+--|
// 12 13 14 15|11|12|13|14|15|11 12 13 14
//            |--+--+--+--+--|
// 17 18 19 20|16|17|18|19|20|16 17 18 19
//            |--+--+--+--+--|
// 22 23 24 25|21|22|23|24|25|21 22 23 24
//            '--'--'--'--'--'
//** Basic Conditions for Rows and Columns: C=10 **
//  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 of the Object: C=10 **
//  n1+n7+n13+n19+n25=C  ... pd1  | n1+n10+n14+n18+n22=C ... pd6
//  n2+n8+n14+n20+n21=C  ... pd2  | n2+n6+n15+n19+n23=C  ... pd7
//  n3+n9+n15+n16+n22=C  ... pd3  | n3+n7+n11+n20+n24=C  ... pd8
//  n4+n10+n11+n17+n23=C ... pd4  | n4+n8+n12+n16+n25=C  ... pd9
//  n5+n6+n12+n18+n24=C  ... pd5  | n5+n9+n13+n17+n21=C  ... pd10
//
```

How can we write our program to make those Latin Units, then?

We must prepare so many as 20x5 flag counters to check if each number of {0, 1, 2, 3, 4} is used strictly once in each row, column and pan-diagonal. If they follow the Latin Condition, every sum of them is consequently equal to the constant sum 10.

We must also count how many times in all each number is used in each unit, and check whether it is used **five times** or not. More or less often it must not be used.

Let me show you the new program for the first object. It is full of my inventions.

```
//** Pan-diagonal Magic Squares of Order 5: Reconstructed by **
//** the 5-th Increment Number System and New Euler's Method **
//** File: 'CES55PDS.c': Dictated by Kanji Setsuda **
//**  in 2003, 2006, 2011, and Revised on Apr.15, 2012 **
//**  Working with MacOSX 10.7.3 & Xcode 4.3.2  **
//
#include <stdio.h>
//
short int cnt, lcnt, cnt2;
long u1,u2;
short cnt3;
short SC, LSM;
short nm[26], uflg[26], tn[26];
short anm[9][27];
char tlu[241][29];
short solt[3601][28];
short cs[4][21];
//
short rw1[5],  cl1[5],  pd1[5],  pb1[5];
short rw2[5],  cl2[5],  pd2[5],  pb2[5];
short rw3[5],  cl3[5],  pd3[5],  pb3[5];
short rw4[5],  cl4[5],  pd4[5],  pb4[5];
short rw5[5],  cl5[5],  pd5[5],  pb5[5];
//
void stp01(void), stp02(void), stp03(void), stp04(void), stp05(void);
void stp06(void), stp07(void), stp08(void), stp09(void), stp10(void);
void stp11(void), stp12(void), stp13(void), stp14(void), stp15(void);
void stp16(void), stp17(void), stp18(void), stp19(void), stp20(void);
void stp21(void), stp22(void), stp23(void), stp24(void), stp25(void);
void lurecord(void), prlunit(short x);
//
short chkmtc(short x, short y);
void cmbcmp(void), solsave(void);
void prans(short x), pr2ans(void), pr1ans(void);
void sortlst(short x), exc(short x);
void chksms(short x);
//
//* Main Program *
int main(void){
 short n;
  printf("\n");
  printf("** Complete Euler Squares 5x5 for the Pan-diagonal Type: Reconstructed\n");
  printf("    by the 5-th Increment Number System and New Euler's Method **\n");
  for(n=0;n<26;n++){nm[n]=0;}
  for(n=0;n<5;n++){uflg[n]=0;
    rw1[n]=0;  cl1[n]=0;  pd1[n]=0;  pb1[n]=0;
    rw2[n]=0;  cl2[n]=0;  pd2[n]=0;  pb2[n]=0;
    rw3[n]=0;  cl3[n]=0;  pd3[n]=0;  pb3[n]=0;
    rw4[n]=0;  cl4[n]=0;  pd4[n]=0;  pb4[n]=0;
    rw5[n]=0;  cl5[n]=0;  pd5[n]=0;  pb5[n]=0;
  }
  SC=4; LSM=10; cnt=0;
  printf("\n");
  printf(" ** List of Latin Units made by /D5i **\n");
  stp01();                          // Make the Latin Units
  if(cnt3>0){prlunit(cnt3);}        // Print the Rest One
  lcnt=cnt; printf(" [Count of Latin Units = %d]\n",lcnt);
  printf("\n");
  printf("** Complete Euler Squares of Order 5 for the Pan-diagonal Type:\n");
  printf("    Reconstructed by the 5-th Increment System and New Euler's Method **\n");
  printf(" [Smart List of Standard Solutions]\n");
  printf("  (Latin Units:/High/Low; [math] Sol_Number#; Check_Sums:|Row|Column\\Pd1/Pd2|)\n");
  cnt=0; cnt3=0;
  cmbcmp();                         // Combine and Compose
  sortlst(cnt);                     // Sort the Data for the Smart List
```

```
  prans(72);                        // Print the Solutions with that Interval
  printf(" [Total Count of Solutions = %d]  [OK!]\n",cnt);
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on Apr.15, 2012 with MacOSX 10.7.3 & Xcode 4.3.2 **\n");
  printf("\n");
 return 0;
}
//
// Level #1:
// Set n1{1,1,1,1}
void stp01(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw1[a]==0)&&(cl1[a]==0)&&(pd1[a]==0)&&(pb1[a]==0)){
        nm[1]=a;
        uflg[a]++; rw1[a]=1; cl1[a]=1; pd1[a]=1; pb1[a]=1;
        stp02();
        uflg[a]--; rw1[a]=0; cl1[a]=0; pd1[a]=0; pb1[a]=0;}}
  }
}
// Set n25{5,5,1,4}
void stp02(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw5[a]==0)&&(cl5[a]==0)&&(pd1[a]==0)&&(pb4[a]==0)){
        nm[25]=a; cnt2=0;
        uflg[a]++; rw5[a]=1; cl5[a]=1; pd1[a]=1; pb4[a]=1;
        stp03();
        uflg[a]--; rw5[a]=0; cl5[a]=0; pd1[a]=0; pb4[a]=0;}}
  }
}
// Set n7{2,2,1,3}
void stp03(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw2[a]==0)&&(cl2[a]==0)&&(pd1[a]==0)&&(pb3[a]==0)){
        nm[7]=a; if(nm[1]==0){cnt2=0;}
        uflg[a]++; rw2[a]=1; cl2[a]=1; pd1[a]=1; pb3[a]=1;
        stp04();
        uflg[a]--; rw2[a]=0; cl2[a]=0; pd1[a]=0; pb3[a]=0;}}
  }
}
// Set n19{4,4,1,2}
void stp04(){
  short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw4[a]==0)&&(cl4[a]==0)&&(pd1[a]==0)&&(pb2[a]==0)){
        nm[19]=a;
        uflg[a]++; rw4[a]=1; cl4[a]=1; pd1[a]=1; pb2[a]=1;
        stp05();
        uflg[a]--; rw4[a]=0; cl4[a]=0; pd1[a]=0; pb2[a]=0;}}
  }
}
// Set n13{3,3,1,5}
void stp05(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw3[a]==0)&&(cl3[a]==0)&&(pd1[a]==0)&&(pb5[a]==0)){
        nm[13]=a;
        uflg[a]++; rw3[a]=1; cl3[a]=1; pd1[a]=1; pb5[a]=1;
```

```
        stp06();
        uflg[a]--; rw3[a]=0; cl3[a]=0; pd1[a]=0; pb5[a]=0;}}
  }
}
// Level #2:
// Set n2{1,2,2,2}
void stp06(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw1[a]==0)&&(cl2[a]==0)&&(pd2[a]==0)&&(pb2[a]==0)){
        nm[2]=a;
        uflg[a]++; rw1[a]=1; cl2[a]=1; pd2[a]=1; pb2[a]=1;
        stp07();
        uflg[a]--; rw1[a]=0; cl2[a]=0; pd2[a]=0; pb2[a]=0;}}
  }
}
// Set n3{1,3,3,3}
void stp07(){
  short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw1[a]==0)&&(cl3[a]==0)&&(pd3[a]==0)&&(pb3[a]==0)){
        nm[3]=a;
        uflg[a]++; rw1[a]=1; cl3[a]=1; pd3[a]=1; pb3[a]=1;
        stp08();
        uflg[a]--; rw1[a]=0; cl3[a]=0; pd3[a]=0; pb3[a]=0;}}
  }
}
// Set n4{1,4,4,4}
void stp08(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw1[a]==0)&&(cl4[a]==0)&&(pd4[a]==0)&&(pb4[a]==0)){
        nm[4]=a;
        uflg[a]++; rw1[a]=1; cl4[a]=1; pd4[a]=1; pb4[a]=1;
        stp09();
        uflg[a]--; rw1[a]=0; cl4[a]=0; pd4[a]=0; pb4[a]=0;}}
  }
}
// Set n5{1,5,5,5}
void stp09(){
  short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw1[a]==0)&&(cl5[a]==0)&&(pd5[a]==0)&&(pb5[a]==0)){
        nm[5]=a;
        uflg[a]++; rw1[a]=1; cl5[a]=1; pd5[a]=1; pb5[a]=1;
        stp10();
        uflg[a]--; rw1[a]=0; cl5[a]=0; pd5[a]=0; pb5[a]=0;}}
  }
}
// Level #3:
// Set n11{3,1,4,3}
void stp10(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw3[a]==0)&&(cl1[a]==0)&&(pd4[a]==0)&&(pb3[a]==0)){
        nm[11]=a;
        uflg[a]++; rw3[a]=1; cl1[a]=1; pd4[a]=1; pb3[a]=1;
        stp11();
        uflg[a]--; rw3[a]=0; cl1[a]=0; pd4[a]=0; pb3[a]=0;}}
  }
```

```
}
// Set n16{4,1,3,4}
void stp11(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw4[a]==0)&&(cl1[a]==0)&&(pd3[a]==0)&&(pb4[a]==0)){
        nm[16]=a;
        uflg[a]++; rw4[a]=1; cl1[a]=1; pd3[a]=1; pb4[a]=1;
        stp12();
        uflg[a]--; rw4[a]=0; cl1[a]=0; pd3[a]=0; pb4[a]=0;}}
  }
}
// Set n21{5,1,2,5}
void stp12(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw5[a]==0)&&(cl1[a]==0)&&(pd2[a]==0)&&(pb5[a]==0)){
        nm[21]=a;
        uflg[a]++; rw5[a]=1; cl1[a]=1; pd2[a]=1; pb5[a]=1;
        stp13();
        uflg[a]--; rw5[a]=0; cl1[a]=0; pd2[a]=0; pb5[a]=0;}}
  }
}
// Set n6{2,1,5,2}
void stp13(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw2[a]==0)&&(cl1[a]==0)&&(pd5[a]==0)&&(pb2[a]==0)){
        nm[6]=a;
        uflg[a]++; rw2[a]=1; cl1[a]=1; pd5[a]=1; pb2[a]=1;
        stp14();
        uflg[a]--; rw2[a]=0; cl1[a]=0; pd5[a]=0; pb2[a]=0;}}
  }
}
// Level #4:
// Set n8{2,3,2,4}
void stp14(){
  short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw2[a]==0)&&(cl3[a]==0)&&(pd2[a]==0)&&(pb4[a]==0)){
        nm[8]=a;
        uflg[a]++; rw2[a]=1; cl3[a]=1; pd2[a]=1; pb4[a]=1;
        stp15();
        uflg[a]--; rw2[a]=0; cl3[a]=0; pd2[a]=0; pb4[a]=0;}}
  }
}
// Set n9{2,4,3,5}
void stp15(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw2[a]==0)&&(cl4[a]==0)&&(pd3[a]==0)&&(pb5[a]==0)){
        nm[9]=a;
        uflg[a]++; rw2[a]=1; cl4[a]=1; pd3[a]=1; pb5[a]=1;
        stp16();
        uflg[a]--; rw2[a]=0; cl4[a]=0; pd3[a]=0; pb5[a]=0;}}
  }
}
// Set n10{2,5,4,1}
void stp16(){
 short a;
```

```
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw2[a]==0)&&(cl5[a]==0)&&(pd4[a]==0)&&(pb1[a]==0)){
        nm[10]=a;
        uflg[a]++; rw2[a]=1; cl5[a]=1; pd4[a]=1; pb1[a]=1;
        stp17();
        uflg[a]--; rw2[a]=0; cl5[a]=0; pd4[a]=0; pb1[a]=0;}}
  }
}
// Set n15{3,5,3,2}
void stp17(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw3[a]==0)&&(cl5[a]==0)&&(pd3[a]==0)&&(pb2[a]==0)){
        nm[15]=a;
        uflg[a]++; rw3[a]=1; cl5[a]=1; pd3[a]=1; pb2[a]=1;
        stp18();
        uflg[a]--; rw3[a]=0; cl5[a]=0; pd3[a]=0; pb2[a]=0;}}
  }
}
// Set n14{3,4,2,1}
void stp18(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw3[a]==0)&&(cl4[a]==0)&&(pd2[a]==0)&&(pb1[a]==0)){
        nm[14]=a;
        uflg[a]++; rw3[a]=1; cl4[a]=1; pd2[a]=1; pb1[a]=1;
        stp19();
        uflg[a]--; rw3[a]=0; cl4[a]=0; pd2[a]=0; pb1[a]=0;}}
  }
}
// Set n12{3,2,5,4}
void stp19(){
 short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw3[a]==0)&&(cl2[a]==0)&&(pd5[a]==0)&&(pb4[a]==0)){
        nm[12]=a;
        uflg[a]++; rw3[a]=1; cl2[a]=1; pd5[a]=1; pb4[a]=1;
        stp20();
        uflg[a]--; rw3[a]=0; cl2[a]=0; pd5[a]=0; pb4[a]=0;}}
  }
}
// Set n18{4,3,5,1}
void stp20(){
  short a;
  for(a=4;a>=0;a--){
    if(uflg[a]<5){
      if((rw4[a]==0)&&(cl3[a]==0)&&(pd5[a]==0)&&(pb1[a]==0)){
        nm[18]=a;
        uflg[a]++; rw4[a]=1; cl3[a]=1; pd5[a]=1; pb1[a]=1;
        stp21();
        uflg[a]--; rw4[a]=0; cl3[a]=0; pd5[a]=0; pb1[a]=0;}}
  }
}
// Set n17{4,2,4,5}
void stp21(){
 short a;
  for(a=0;a<5;a++){
    if(uflg[a]<5){
      if((rw4[a]==0)&&(cl2[a]==0)&&(pd4[a]==0)&&(pb5[a]==0)){
        nm[17]=a;
        uflg[a]++; rw4[a]=1; cl2[a]=1; pd4[a]=1; pb5[a]=1;
```

```
          stp22();
          uflg[a]--; rw4[a]=0; cl2[a]=0; pd4[a]=0; pb5[a]=0;}}
    }
}
// Set n20{4,5,2,3}
void stp22(){
 short a;
   for(a=0;a<5;a++){
     if(uflg[a]<5){
       if((rw4[a]==0)&&(cl5[a]==0)&&(pd2[a]==0)&&(pb3[a]==0)){
         nm[20]=a;
         uflg[a]++; rw4[a]=1; cl5[a]=1; pd2[a]=1; pb3[a]=1;
         stp23();
         uflg[a]--; rw4[a]=0; cl5[a]=0; pd2[a]=0; pb3[a]=0;}}
   }
}
// Set n24{5,4,5,3}
void stp23(){
 short a;
   for(a=0;a<5;a++){
     if(uflg[a]<5){
       if((rw5[a]==0)&&(cl4[a]==0)&&(pd5[a]==0)&&(pb3[a]==0)){
         nm[24]=a;
         uflg[a]++; rw5[a]=1; cl4[a]=1; pd5[a]=1; pb3[a]=1;
         stp24();
         uflg[a]--; rw5[a]=0; cl4[a]=0; pd5[a]=0; pb3[a]=0;}}
   }
}
// Set n23{5,3,4,2}
void stp24(){
 short a;
   for(a=0;a<5;a++){
     if(uflg[a]<5){
       if((rw5[a]==0)&&(cl3[a]==0)&&(pd4[a]==0)&&(pb2[a]==0)){
         nm[23]=a;
         uflg[a]++; rw5[a]=1; cl3[a]=1; pd4[a]=1; pb2[a]=1;
         stp25();
         uflg[a]--; rw5[a]=0; cl3[a]=0; pd4[a]=0; pb2[a]=0;}}
   }
}
// Set n22{5,2,3,1}
void stp25(){
 short a;
   for(a=4;a>=0;a--){
     if(uflg[a]<5){
       if((rw5[a]==0)&&(cl2[a]==0)&&(pd3[a]==0)&&(pb1[a]==0)){
         nm[22]=a;
         uflg[a]++; rw5[a]=1; cl2[a]=1; pd3[a]=1; pb1[a]=1;
         lurecord();
         uflg[a]--; rw5[a]=0; cl2[a]=0; pd3[a]=0; pb1[a]=0;}}
   }
}
//
// Record Latin Units
void lurecord(){
 short n;
   tlu[cnt][0]=cnt+1;
   for(n=1;n<26;n++){tlu[cnt][n]=nm[n];}
   cnt++; cnt2++;
   if(cnt2==1){anm[cnt3][0]=cnt;
     for(n=1;n<26;n++){anm[cnt3][n]=nm[n];}
     cnt3++; if(cnt3==7){prlunit(cnt3); cnt3=0;}
   }
}
//
```

```
// Print Latin Units
void prlunit(short x){
 short l,m,m5,n;
  printf("");
  for(l=0;l<x;l++){printf("  %9d/",anm[l][0]);}
  printf("\n");
  for(m=0;m<5;m++){m5=m*5;
    printf("");
    for(l=0;l<x;l++){printf("  ");
      for(n=1;n<6;n++){printf("%2d",anm[l][m5+n]);}}
    printf("\n");
  }
}
//
// Check how well they match
short chkmtc(short x, short y){
 short n,d1,d2;
  d1=0; for(n=1;n<26;n++){if(tlu[x][n]==tlu[y][n]){d1++;}}
  d2=0; for(n=1;n<26;n++){if(tlu[x][n]+tlu[y][n]==SC){d2++;}}
  if(d1<d2){d1=d2;}
 return d1;
}
//
//* Combine, Compose and Print *
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<(lcnt/2);u1++){
    for(u2=0;u2<lcnt;u2++){
      if(chkmtc(u1,u2)==5){
        for(n=1;n<26;n++){uflg[n]=0;}
        for(n=1;n<26;n++){
          d=tlu[u1][n]*5+tlu[u2][n]+1;
          nm[n]=d; uflg[d]++;
        }
        fc=0;
        for(n=1;n<26;n++){if(uflg[n]==0){fc++;}}
        if(fc==0){
          if((nm[1]<nm[5])&&(nm[5]<nm[21])&&(nm[1]<nm[25])){solsave();}}
      }
    }}
}
//
// Save Answers to the Solution List
void solsave(){
 short n;
  solt[cnt][0]=cnt+1;
  for(n=1;n<26;n++){solt[cnt][n]=nm[n];}
  solt[cnt][26]=u1;  solt[cnt][27]=u2;
  cnt++;
}
//
// Print the Solution List
void prans(short x){
 short s,lu1,lu2;
 short n;
  for(s=0;s<cnt;s=s+x){
    anm[0][0]=s+1; lu1=solt[s][26]; lu2=solt[s][27];
    for(n=1;n<26;n++){anm[0][n]=solt[s][n];}
    for(n=1;n<26;n++){tn[n]=solt[s][n]-1; anm[1][n]=tn[n];}; chksms(1);
    for(n=1;n<26;n++){anm[2][n]=tlu[lu1][n];}
    for(n=1;n<26;n++){anm[3][n]=tlu[lu2][n];}
    anm[2][0]=lu1+1; anm[3][0]=lu2+1;
    pr1ans();
  }
}
```

```
//
// Print the Rest One
void pr1ans(){
 short l,l5,n;
  printf("  Lu:%4d/H%10d/L",anm[2][0],anm[3][0]);
  printf("   [Math]        |Rw|Cl\\P1/P2|%15d#\n",anm[0][0]);
  for(l=0;l<5;l++){l5=l*5;
    printf(" ");
    for(n=1;n<6;n++){printf("%2d",anm[2][l5+n]);}; printf("  ");
    for(n=1;n<6;n++){printf("%2d",anm[3][l5+n]);}; printf("  ");
    for(n=1;n<6;n++){printf("%3d",anm[1][l5+n]);}
    printf("|%2d|%2d|%2d|%2d| ",cs[1][l+1],cs[1][l+6],cs[1][l+11],cs[1][l+16]);
    for(n=1;n<6;n++){printf("%3d",anm[0][l5+n]);}; printf("\n");
  }
}
//
// Sort the Solution List
void sortlst(short x){
 short mx,m,f;
 short cd1,cd2,cd3,cd4,cd5,cd6,cd7,cd8;
  mx=x-1;
  do{f=0;
    for(m=0;m<mx;m++){
      cd1=solt[m][1]; cd2=solt[m+1][1];
      cd3=solt[m][25]; cd4=solt[m+1][25];
      cd5=solt[m][7]*25+solt[m][9]; cd6=solt[m+1][7]*25+solt[m+1][9];
      cd7=solt[m][5]*25+solt[m][21]; cd8=solt[m+1][5]*25+solt[m+1][21];
      if(cd1>cd2){exc(m); f=1;}
      else if((cd1==cd2)&&(cd3<cd4)){exc(m); f=1;}
      else if((cd1==cd2)&&(cd3==cd4)&&(cd5>cd6)){exc(m); f=1;}
      else if((cd1==cd2)&&(cd3==cd4)&&(cd5==cd6)&&(cd7>cd8)){exc(m); f=1;}
    }
    mx--;
  }while(f>0);
}
//
void exc(short x){
 short d,n;
  for(n=0;n<28;n++){
    d=solt[x][n]; solt[x][n]=solt[x+1][n]; solt[x+1][n]=d;}
}
//
//* Check Sums *
void chksms(short x){
  cs[x][1]=tn[1]+tn[2]+tn[3]+tn[4]+tn[5];
  cs[x][2]=tn[6]+tn[7]+tn[8]+tn[9]+tn[10];
  cs[x][3]=tn[11]+tn[12]+tn[13]+tn[14]+tn[15];
  cs[x][4]=tn[16]+tn[17]+tn[18]+tn[19]+tn[20];
  cs[x][5]=tn[21]+tn[22]+tn[23]+tn[24]+tn[25];
  cs[x][6]=tn[1]+tn[6]+tn[11]+tn[16]+tn[21];
  cs[x][7]=tn[2]+tn[7]+tn[12]+tn[17]+tn[22];
  cs[x][8]=tn[3]+tn[8]+tn[13]+tn[18]+tn[23];
  cs[x][9]=tn[4]+tn[9]+tn[14]+tn[19]+tn[24];
  cs[x][10]=tn[5]+tn[10]+tn[15]+tn[20]+tn[25];
  cs[x][11]=tn[1]+tn[7]+tn[13]+tn[19]+tn[25];
  cs[x][12]=tn[2]+tn[8]+tn[14]+tn[20]+tn[21];
  cs[x][13]=tn[3]+tn[9]+tn[15]+tn[16]+tn[22];
  cs[x][14]=tn[4]+tn[10]+tn[11]+tn[17]+tn[23];
  cs[x][15]=tn[5]+tn[6]+tn[12]+tn[18]+tn[24];
  cs[x][16]=tn[1]+tn[10]+tn[14]+tn[18]+tn[22];
  cs[x][17]=tn[2]+tn[6]+tn[15]+tn[19]+tn[23];
  cs[x][18]=tn[3]+tn[7]+tn[11]+tn[20]+tn[24];
  cs[x][19]=tn[4]+tn[8]+tn[12]+tn[16]+tn[25];
  cs[x][20]=tn[5]+tn[9]+tn[13]+tn[17]+tn[21];
}
```

The next list shows the total result of my recent calculation:

```
** Complete Euler Squares 5x5 for the Pan-diagonal Type: Reconstructed
    by the 5-th Increment Number System and New Euler's Method **

 ** List of Latin Units made by /D5i **
        1/          5/          9/         13/         17/         21/         25/
  0 4 3 2 1   0 4 3 1 2   0 4 2 1 3   0 3 4 2 1   0 3 4 1 2   0 3 2 1 4   0 3 2 1 4
  2 1 0 4 3   1 2 0 4 3   1 3 0 4 2   2 1 0 3 4   1 2 0 3 4   1 4 0 3 2   2 1 4 0 3
  4 3 2 1 0   4 3 1 2 0   4 2 1 3 0   3 4 2 1 0   3 4 1 2 0   3 2 1 4 0   4 0 3 2 1
  1 0 4 3 2   2 0 4 3 1   3 0 4 2 1   1 0 3 4 2   2 0 3 4 1   4 0 3 2 1   3 2 1 4 0
  3 2 1 0 4   3 1 2 0 4   2 1 3 0 4   4 2 1 0 3   4 1 2 0 3   2 1 4 0 3   1 4 0 3 2
        29/         33/         37/         41/         45/         49/         61/
  0 2 4 1 3   0 2 3 1 4   0 3 1 2 4   0 2 1 3 4   0 2 1 4 3   1 4 3 2 0   1 3 4 2 0
  1 3 0 2 4   1 4 0 2 3   1 2 4 0 3   1 3 4 0 2   1 4 3 0 2   2 0 1 4 3   2 0 1 3 4
  2 4 1 3 0   2 3 1 4 0   4 0 3 1 2   4 0 2 1 3   3 0 2 1 4   4 3 2 0 1   3 4 2 0 1
  3 0 2 4 1   4 0 2 3 1   3 1 2 4 0   2 1 3 4 0   2 1 4 3 0   0 1 4 3 2   0 1 3 4 2
  4 1 3 0 2   3 1 4 0 2   2 4 0 3 1   3 4 0 2 1   4 3 0 2 1   3 2 0 1 4   4 2 0 1 3
        73/         85/         97/        109/        121/        133/        145/
  1 3 2 0 4   1 3 0 2 4   2 4 3 1 0   2 3 4 1 0   2 3 1 0 4   2 3 0 1 4   3 4 2 1 0
  2 0 4 1 3   0 2 4 1 3   1 0 2 4 3   1 0 2 3 4   1 0 4 2 3   0 1 4 2 3   1 0 3 4 2
  4 1 3 2 0   4 1 3 0 2   4 3 1 0 2   3 4 1 0 2   4 2 3 1 0   4 2 3 0 1   4 2 1 0 3
  3 2 0 4 1   3 0 2 4 1   0 2 4 3 1   0 2 3 4 1   3 1 0 4 2   3 0 1 4 2   0 3 4 2 1
  0 4 1 3 2   2 4 1 3 0   3 1 0 2 4   4 1 0 2 3   0 4 2 3 1   1 4 2 3 0   2 1 0 3 4
       157/        169/        181/        193/        205/        217/        229/
  3 2 4 1 0   3 2 1 0 4   3 2 0 1 4   4 3 2 1 0   4 2 3 1 0   4 2 1 0 3   4 2 0 1 3
  1 0 3 2 4   1 0 4 3 2   0 1 4 3 2   1 0 4 3 2   1 0 4 2 3   1 0 3 4 2   0 1 3 4 2
  2 4 1 0 3   4 3 2 1 0   4 3 2 0 1   3 2 1 0 4   2 3 1 0 4   3 4 2 1 0   3 4 2 0 1
  0 3 2 4 1   2 1 0 4 3   2 0 1 4 3   0 4 3 2 1   0 4 2 3 1   2 1 0 3 4   2 0 1 3 4
  4 1 0 3 2   0 4 3 2 1   1 4 3 2 0   2 1 0 4 3   3 1 0 4 2   0 3 4 2 1   1 3 4 2 0
 [Count of Latin Units = 240]
```

Let me skip a little in the way and come to my conclusion. The next list shows the abstract set of our reconstructions in the smarter style in the sorted order.

```
** Complete Euler Squares of Order 5 for the Pan-diagonal Type:
    Reconstructed by the 5-th Increment System and New Euler's Method **
  [Smart List of Standard Solutions]
  (Latin Units:/High/Low; [Math] Check_Sums:|Row|Column\Pd1/Pd2|; Sol_Number#)
  Lu:   3/H         4/L   [Math]         |Rw|Cl\P1/P2|              1#
  0 4 2 3 1   0 3 4 1 2    0 23 14 16  7|60|60|60|60|   1 24 15 17  8
  3 1 0 4 2   4 1 2 0 3   19  6  2 20 13|60|60|60|60|  20  7  3 21 14
  4 2 3 1 0   2 0 3 4 1   22 10 18  9  1|60|60|60|60|  23 11 19 10  2
  1 0 4 2 3   3 4 1 2 0    8  4 21 12 15|60|60|60|60|   9  5 22 13 16
  2 3 1 0 4   1 2 0 3 4   11 17  5  3 24|60|60|60|60|  12 18  6  4 25
  Lu:   3/H        27/L   [Math]         |Rw|Cl\P1/P2|             73#
  0 4 2 3 1   0 4 2 1 3    0 24 12 16  8|60|60|60|60|   1 25 13 17  9
  3 1 0 4 2   2 1 3 0 4   17  6  3 20 14|60|60|60|60|  18  7  4 21 15
  4 2 3 1 0   3 0 4 2 1   23 10 19  7  1|60|60|60|60|  24 11 20  8  2
  1 0 4 2 3   4 2 1 3 0    9  2 21 13 15|60|60|60|60|  10  3 22 14 16
  2 3 1 0 4   1 3 0 4 2   11 18  5  4 22|60|60|60|60|  12 19  6  5 23
  Lu:  16/H         4/L   [Math]         |Rw|Cl\P1/P2|            145#
  0 3 2 4 1   0 3 4 1 2    0 18 14 21  7|60|60|60|60|   1 19 15 22  8
  4 1 0 3 2   4 1 2 0 3   24  6  2 15 13|60|60|60|60|  25  7  3 16 14
  3 2 4 1 0   2 0 3 4 1   17 10 23  9  1|60|60|60|60|  18 11 24 10  2
  1 0 3 2 4   3 4 1 2 0    8  4 16 12 20|60|60|60|60|   9  5 17 13 21
  2 4 1 0 3   1 2 0 3 4   11 22  5  3 19|60|60|60|60|  12 23  6  4 20
```

```
Lu:  16/H        27/L   [Math]        |Rw|Cl\P1/P2|            217#
0 3 2 4 1   0 4 2 1 3    0 19 12 21  8|60|60|60|60|   1 20 13 22  9
4 1 0 3 2   2 1 3 0 4   22  6  3 15 14|60|60|60|60|  23  7  4 16 15
3 2 4 1 0   3 0 4 2 1   18 10 24  7  1|60|60|60|60|  19 11 25  8  2
1 0 3 2 4   4 2 1 3 0    9  2 16 13 20|60|60|60|60|  10  3 17 14 21
2 4 1 0 3   1 3 0 4 2   11 23  5  4 17|60|60|60|60|  12 24  6  5 18

Lu:  28/H        4/L   [Math]        |Rw|Cl\P1/P2|            289#
0 2 3 4 1   0 3 4 1 2    0 13 19 21  7|60|60|60|60|   1 14 20 22  8
4 1 0 2 3   4 1 2 0 3   24  6  2 10 18|60|60|60|60|  25  7  3 11 19
2 3 4 1 0   2 0 3 4 1   12 15 23  9  1|60|60|60|60|  13 16 24 10  2
1 0 2 3 4   3 4 1 2 0    8  4 11 17 20|60|60|60|60|   9  5 12 18 21
3 4 1 0 2   1 2 0 3 4   16 22  5  3 14|60|60|60|60|  17 23  6  4 15

Lu:  28/H        27/L   [Math]        |Rw|Cl\P1/P2|            361#
0 2 3 4 1   0 4 2 1 3    0 14 17 21  8|60|60|60|60|   1 15 18 22  9
4 1 0 2 3   2 1 3 0 4   22  6  3 10 19|60|60|60|60|  23  7  4 11 20
2 3 4 1 0   3 0 4 2 1   13 15 24  7  1|60|60|60|60|  14 16 25  8  2
1 0 2 3 4   4 2 1 3 0    9  2 11 18 20|60|60|60|60|  10  3 12 19 21
3 4 1 0 2   1 3 0 4 2   16 23  5  4 12|60|60|60|60|  17 24  6  5 13

Lu:  40/H        4/L   [Math]        |Rw|Cl\P1/P2|            433#
0 1 3 4 2   0 3 4 1 2    0  8 19 21 12|60|60|60|60|   1  9 20 22 13
4 2 0 1 3   4 1 2 0 3   24 11  2  5 18|60|60|60|60|  25 12  3  6 19
1 3 4 2 0   2 0 3 4 1    7 15 23 14  1|60|60|60|60|   8 16 24 15  2
2 0 1 3 4   3 4 1 2 0   13  4  6 17 20|60|60|60|60|  14  5  7 18 21
3 4 2 0 1   1 2 0 3 4   16 22 10  3  9|60|60|60|60|  17 23 11  4 10

Lu:  40/H        27/L   [Math]        |Rw|Cl\P1/P2|            505#
0 1 3 4 2   0 4 2 1 3    0  9 17 21 13|60|60|60|60|   1 10 18 22 14
4 2 0 1 3   2 1 3 0 4   22 11  3  5 19|60|60|60|60|  23 12  4  6 20
1 3 4 2 0   3 0 4 2 1    8 15 24 12  1|60|60|60|60|   9 16 25 13  2
2 0 1 3 4   4 2 1 3 0   14  2  6 18 20|60|60|60|60|  15  3  7 19 21
3 4 2 0 1   1 3 0 4 2   16 23 10  4  7|60|60|60|60|  17 24 11  5  8

Lu:   3/H        52/L   [Math]        |Rw|Cl\P1/P2|            577#
0 4 2 3 1   1 3 4 0 2    1 23 14 15  7|60|60|60|60|   2 24 15 16  8
3 1 0 4 2   4 0 2 1 3   19  5  2 21 13|60|60|60|60|  20  6  3 22 14
4 2 3 1 0   2 1 3 4 0   22 11 18  9  0|60|60|60|60|  23 12 19 10  1
1 0 4 2 3   3 4 0 2 1    8  4 20 12 16|60|60|60|60|   9  5 21 13 17
2 3 1 0 4   0 2 1 3 4   10 17  6  3 24|60|60|60|60|  11 18  7  4 25

Lu:  28/H        52/L   [Math]        |Rw|Cl\P1/P2|            865#
0 2 3 4 1   1 3 4 0 2    1 13 19 20  7|60|60|60|60|   2 14 20 21  8
4 1 0 2 3   4 0 2 1 3   24  5  2 11 18|60|60|60|60|  25  6  3 12 19
2 3 4 1 0   2 1 3 4 0   12 16 23  9  0|60|60|60|60|  13 17 24 10  1
1 0 2 3 4   3 4 0 2 1    8  4 10 17 21|60|60|60|60|   9  5 11 18 22
3 4 1 0 2   0 2 1 3 4   15 22  6  3 14|60|60|60|60|  16 23  7  4 15

Lu:   3/H       100/L   [Math]        |Rw|Cl\P1/P2|           1153#
0 4 2 3 1   2 3 4 0 1    2 23 14 15  6|60|60|60|60|   3 24 15 16  7
3 1 0 4 2   4 0 1 2 3   19  5  1 22 13|60|60|60|60|  20  6  2 23 14
4 2 3 1 0   1 2 3 4 0   21 12 18  9  0|60|60|60|60|  22 13 19 10  1
1 0 4 2 3   3 4 0 1 2    8  4 20 11 17|60|60|60|60|   9  5 21 12 18
2 3 1 0 4   0 1 2 3 4   10 16  7  3 24|60|60|60|60|  11 17  8  4 25

Lu:  28/H       100/L   [Math]        |Rw|Cl\P1/P2|           1441#
0 2 3 4 1   2 3 4 0 1    2 13 19 20  6|60|60|60|60|   3 14 20 21  7
4 1 0 2 3   4 0 1 2 3   24  5  1 12 18|60|60|60|60|  25  6  2 13 19
2 3 4 1 0   1 2 3 4 0   11 17 23  9  0|60|60|60|60|  12 18 24 10  1
1 0 2 3 4   3 4 0 1 2    8  4 10 16 22|60|60|60|60|   9  5 11 17 23
3 4 1 0 2   0 1 2 3 4   15 21  7  3 14|60|60|60|60|  16 22  8  4 15
```

```
Lu:   3/H       148/L   [Math]        |Rw|Cl\P1/P2|           1729#
0 4 2 3 1   3 2 4 0 1    3 22 14 15  6|60|60|60|60|   4 23 15 16  7
3 1 0 4 2   4 0 1 3 2   19  5  1 23 12|60|60|60|60|  20  6  2 24 13
4 2 3 1 0   1 3 2 4 0   21 13 17  9  0|60|60|60|60|  22 14 18 10  1
1 0 4 2 3   2 4 0 1 3    7  4 20 11 18|60|60|60|60|   8  5 21 12 19
2 3 1 0 4   0 1 3 2 4   10 16  8  2 24|60|60|60|60|  11 17  9  3 25

Lu:  28/H       148/L   [Math]        |Rw|Cl\P1/P2|           2017#
0 2 3 4 1   3 2 4 0 1    3 12 19 20  6|60|60|60|60|   4 13 20 21  7
4 1 0 2 3   4 0 1 3 2   24  5  1 13 17|60|60|60|60|  25  6  2 14 18
2 3 4 1 0   1 3 2 4 0   11 18 22  9  0|60|60|60|60|  12 19 23 10  1
1 0 2 3 4   2 4 0 1 3    7  4 10 16 23|60|60|60|60|   8  5 11 17 24
3 4 1 0 2   0 1 3 2 4   15 21  8  2 14|60|60|60|60|  16 22  9  3 15

Lu:   3/H       196/L   [Math]        |Rw|Cl\P1/P2|           2305#
0 4 2 3 1   4 2 3 0 1    4 22 13 15  6|60|60|60|60|   5 23 14 16  7
3 1 0 4 2   3 0 1 4 2   18  5  1 24 12|60|60|60|60|  19  6  2 25 13
4 2 3 1 0   1 4 2 3 0   21 14 17  8  0|60|60|60|60|  22 15 18  9  1
1 0 4 2 3   2 3 0 1 4    7  3 20 11 19|60|60|60|60|   8  4 21 12 20
2 3 1 0 4   0 1 4 2 3   10 16  9  2 23|60|60|60|60|  11 17 10  3 24

Lu:  28/H       196/L   [Math]        |Rw|Cl\P1/P2|           2593#
0 2 3 4 1   4 2 3 0 1    4 12 18 20  6|60|60|60|60|   5 13 19 21  7
4 1 0 2 3   3 0 1 4 2   23  5  1 14 17|60|60|60|60|  24  6  2 15 18
2 3 4 1 0   1 4 2 3 0   11 19 22  8  0|60|60|60|60|  12 20 23  9  1
1 0 2 3 4   2 3 0 1 4    7  3 10 16 24|60|60|60|60|   8  4 11 17 25
3 4 1 0 2   0 1 4 2 3   15 21  9  2 13|60|60|60|60|  16 22 10  3 14

Lu:  53/H         4/L   [Math]        |Rw|Cl\P1/P2|           2881#
1 4 3 0 2   0 3 4 1 2    5 23 19  1 12|60|60|60|60|   6 24 20  2 13
0 2 1 4 3   4 1 2 0 3    4 11  7 20 18|60|60|60|60|   5 12  8 21 19
4 3 0 2 1   2 0 3 4 1   22 15  3 14  6|60|60|60|60|  23 16  4 15  7
2 1 4 3 0   3 4 1 2 0   13  9 21 17  0|60|60|60|60|  14 10 22 18  1
3 0 2 1 4   1 2 0 3 4   16  2 10  8 24|60|60|60|60|  17  3 11  9 25

  53/ 50/ 3025#     53/ 98/ 3169#     53/146/ 3313#     53/194/ 3457#
 7 23 20  1 14     8 22 20  1 14     9 22 20  1 13    10 22 19  1 13
 5 11  9 22 18     5 11  9 23 17     5 11  8 24 17     4 11  8 25 17
24 17  3 15  6    24 18  2 15  6    23 19  2 15  6    23 20  2 14  6
13 10 21 19  2    12 10 21 19  3    12 10 21 18  4    12  9 21 18  5
16  4 12  8 25    16  4 13  7 25    16  3 14  7 25    16  3 15  7 24   . . . . .

 [Total Count of Solutions = 3600]  [OK!]
** Calculated and Listed by Kanji Setsuda
    on Apr.15, 2012 with MacOSX 10.7.3 & Xcode 4.3.2 **
```

These 3600 solutions reconstructed are just the same with the ones we got before. We could neither make any other type of solutions, nor miss any necessary one. They also verified the result of another method “Knight’s Tour Composition”.

I am happy to report that this ‘Greco-Latinian Method’ could run very quick. It really took only one minute to make all of these 3600 solutions, though it once took more than 30 minutes to make the same set of solutions by our ordinary method.

New Euler’s Method, or Greco-Latinian Method, showing its high speed, should really give us a good relief for our hardest research of high orders.

## 4. How to Make Simultaneous MS55: Self-complementary and Pan-diagonal

I want to report briefly about the 16 Simultaneous MS55, “Suzuki Squares”, making them by applying the same method with the 5th Increment Number System.

The next lists show (1) the definitions of Latin Units, (2) the Program I dictated, but

only the core part of it, and (3) the complete list of its execution result.

```
//
//** Basic Form for the Simultaneous MS55: SC & PD; **
//              .--.--.--.--.--.
//   2  3  4  5| 1| 2| 3| 4| 5| 1  2  3  4
//             |--+--+--+--+--|
//   7  8  9 10| 6| 7| 8| 9|10| 6  7  8  9
//             |--+--+--+--+--|
//  12 13 14 15|11|12|13|14|15|11 12 13 14
//             |--+--+--+--+--|
//  17 18 19 20|16|17|18|19|20|16 17 18 19
//             |--+--+--+--+--|
//  22 23 24 25|21|22|23|24|25|21 22 23 24
//             '--'--'--'--'--'
//** Self-complementary Conditions of the Object: SC=4 **
//  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=SC ... scc
//** Basic Conditions for Rows and Columns: C=10 **
//  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 of the Object: C=10 **
//  n1+n7+n13+n19+n25=C  ... pd1  | n1+n10+n14+n18+n22=C ... pd6
//  n2+n8+n14+n20+n21=C  ... pd2  | n2+n6+n15+n19+n23=C  ... pd7
//  n3+n9+n15+n16+n22=C  ... pd3  | n3+n7+n11+n20+n24=C  ... pd8
//  n4+n10+n11+n17+n23=C ... pd4  | n4+n8+n12+n16+n25=C  ... pd9
//  n5+n6+n12+n18+n24=C  ... pd5  | n5+n9+n13+n17+n21=C  ... pd10
//** List-forming Inequality Conditions for the Standard Solutions **
//   n1<n5; n5<n21; and n1<n25;
//  . . .(skip). . .
//
//* Main Program *
int main(void){
 short n;
  printf("\n");
  printf("** Simultaneous Magic Squares of Order 5: Self-complementary and Pan-diagonal;\n");
  printf("    'Complete Euler Type' Reconstructed by the 'Greco-Latinian Method' **\n");
  for(n=0;n<26;n++){nm[n]=0;}
  for(n=0;n<5;n++){uflg[n]=0;
     rw1[n]=0;  cl1[n]=0;  pd1[n]=0;  pb1[n]=0;
     rw2[n]=0;  cl2[n]=0;  pd2[n]=0;  pb2[n]=0;
     rw3[n]=0;  cl3[n]=0;  pd3[n]=0;  pb3[n]=0;
     rw4[n]=0;  cl4[n]=0;  pd4[n]=0;  pb4[n]=0;
     rw5[n]=0;  cl5[n]=0;  pd5[n]=0;  pb5[n]=0;
  }
  SC=4; LSM=10; cnt=0;
  printf("\n");
  printf(" ** List of Latin Units by /D5i **\n");
  stp00();                        // Make the Latin Units by /D5i
  lcnt=cnt;
  prlunit(8);
  printf("\n");
  printf("** Complete Euler Type of Simultaneous MS55: Self-complementary and Pan-diagonal; **\n");
  printf("  ** Primitive List of Reconstruction with Full Data: Check Sums **\n");
  printf("  [Latin Units: High, Low; [Mathem]; Check Sums:|Row|Column\\Pd1/Pd2|; [S.Nmb]]\n");
  cnt=0;
  cmbcmp();                       // Combine and Compose
  printf(" [Solution Counts = %d]  [OK!]\n",cnt);
  printf("** Calculated and listed by Kanji Setsuda\n");
  printf("    on Apr.15, 2012 with MacOSX 10.7.3 & Xcode 4.2.1 **\n");
  printf("\n");
 return 0;
}
```

```
// Level #0:
// Set n13{3,3,1,5}
void stp00(){
 short a;
  a=2; nm[13]=a;
  uflg[a]++; rw3[a]=1; cl3[a]=1; pd1[a]=1; pb5[a]=1;
  stp01();
  uflg[a]--; rw3[a]=0; cl3[a]=0; pd1[a]=0; pb5[a]=0;
}
// Level #1:/
// Set n1{1,1,1,1} and n25{5,5,1,4}
void stp01(){
 short a,b;
  for(a=0;a<5;a++){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw1[a]==0)&&(cl1[a]==0)&&(pd1[a]==0)&&(pb1[a]==0)){
        if((rw5[b]==0)&&(cl5[b]==0)&&(pd1[b]==0)&&(pb4[b]==0)){
          nm[1]=a; nm[25]=b;
          uflg[a]++; rw1[a]=1; cl1[a]=1; pd1[a]=1; pb1[a]=1;
          uflg[b]++; rw5[b]=1; cl5[b]=1; pd1[b]=1; pb4[b]=1;
          stp02();
          uflg[b]--; rw5[b]=0; cl5[b]=0; pd1[b]=0; pb4[b]=0;
          uflg[a]--; rw1[a]=0; cl1[a]=0; pd1[a]=0; pb1[a]=0;}}}
  }
}
// Set n7{2,2,1,3} and n19{4,4,1,2}
void stp02(){
 short a,b;
  for(a=0;a<5;a++){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw2[a]==0)&&(cl2[a]==0)&&(pd1[a]==0)&&(pb3[a]==0)){
        if((rw4[b]==0)&&(cl4[b]==0)&&(pd1[b]==0)&&(pb2[b]==0)){
          nm[7]=a; nm[19]=b;
          uflg[a]++; rw2[a]=1; cl2[a]=1; pd1[a]=1; pb3[a]=1;
          uflg[b]++; rw4[b]=1; cl4[b]=1; pd1[b]=1; pb2[b]=1;
          stp03();
          uflg[b]--; rw4[b]=0; cl4[b]=0; pd1[b]=0; pb2[b]=0;
          uflg[a]--; rw2[a]=0; cl2[a]=0; pd1[a]=0; pb3[a]=0;}}}
  }
}
// Level #2:
// Set n2{1,2,2,2} and n24{5,4,5,3}
void stp03(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw1[a]==0)&&(cl2[a]==0)&&(pd2[a]==0)&&(pb2[a]==0)){
        if((rw5[b]==0)&&(cl4[b]==0)&&(pd5[b]==0)&&(pb3[b]==0)){
          nm[2]=a; nm[24]=b;
          uflg[a]++; rw1[a]=1; cl2[a]=1; pd2[a]=1; pb2[a]=1;
          uflg[b]++; rw5[b]=1; cl4[b]=1; pd5[b]=1; pb3[b]=1;
          stp04();
          uflg[b]--; rw5[b]=0; cl4[b]=0; pd5[b]=0; pb3[b]=0;
          uflg[a]--; rw1[a]=0; cl2[a]=0; pd2[a]=0; pb2[a]=0;}}}
  }
}
// Set n3{1,3,3,3} and n23{5,3,4,2}
void stp04(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw1[a]==0)&&(cl3[a]==0)&&(pd3[a]==0)&&(pb3[a]==0)){
        if((rw5[b]==0)&&(cl3[b]==0)&&(pd4[b]==0)&&(pb2[b]==0)){
          nm[3]=a; nm[23]=b;
          uflg[a]++; rw1[a]=1; cl3[a]=1; pd3[a]=1; pb3[a]=1;
```

```
            uflg[b]++; rw5[b]=1; cl3[b]=1; pd4[b]=1; pb2[b]=1;
            stp05();
            uflg[b]--; rw5[b]=0; cl3[b]=0; pd4[b]=0; pb2[b]=0;
            uflg[a]--; rw1[a]=0; cl3[a]=0; pd3[a]=0; pb3[a]=0;}}}
  }
}
// Set n4{1,4,4,4} and n22{5,2,3,1}
void stp05(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw1[a]==0)&&(cl4[a]==0)&&(pd4[a]==0)&&(pb4[a]==0)){
        if((rw5[b]==0)&&(cl2[b]==0)&&(pd3[b]==0)&&(pb1[b]==0)){
          nm[4]=a; nm[22]=b;
          uflg[a]++; rw1[a]=1; cl4[a]=1; pd4[a]=1; pb4[a]=1;
          uflg[b]++; rw5[b]=1; cl2[b]=1; pd3[b]=1; pb1[b]=1;
          stp06();
          uflg[b]--; rw5[b]=0; cl2[b]=0; pd3[b]=0; pb1[b]=0;
          uflg[a]--; rw1[a]=0; cl4[a]=0; pd4[a]=0; pb4[a]=0;}}}
  }
}
// Set n5{1,5,5,5} and n21{5,1,2,5}
void stp06(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw1[a]==0)&&(cl5[a]==0)&&(pd5[a]==0)&&(pb5[a]==0)){
        if((rw5[b]==0)&&(cl1[b]==0)&&(pd2[b]==0)&&(pb5[b]==0)){
          nm[5]=a; nm[21]=b;
          uflg[a]++; rw1[a]=1; cl5[a]=1; pd5[a]=1; pb5[a]=1;
          uflg[b]++; rw5[b]=1; cl1[b]=1; pd2[b]=1; pb5[b]=1;
          stp07();
          uflg[b]--; rw5[b]=0; cl1[b]=0; pd2[b]=0; pb5[b]=0;
          uflg[a]--; rw1[a]=0; cl5[a]=0; pd5[a]=0; pb5[a]=0;}}}
  }
}
// Level #3:
//  . . .(skip). . .
//
// Level #4:
// Set n8{2,3,2,4} and n18{4,3,5,1}
void stp10(){
 short a,b;
  for(a=0;a<5;a++){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw2[a]==0)&&(cl3[a]==0)&&(pd2[a]==0)&&(pb4[a]==0)){
        if((rw4[b]==0)&&(cl3[b]==0)&&(pd5[b]==0)&&(pb1[b]==0)){
          nm[8]=a; nm[18]=b;
          uflg[a]++; rw2[a]=1; cl3[a]=1; pd2[a]=1; pb4[a]=1;
          uflg[b]++; rw4[b]=1; cl3[b]=1; pd5[b]=1; pb1[b]=1;
          stp11();
          uflg[b]--; rw4[b]=0; cl3[b]=0; pd5[b]=0; pb1[b]=0;
          uflg[a]--; rw2[a]=0; cl3[a]=0; pd2[a]=0; pb4[a]=0;}}}
  }
}
// Set n9{2,4,3,5} and n17{4,2,4,5}
void stp11(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw2[a]==0)&&(cl4[a]==0)&&(pd3[a]==0)&&(pb5[a]==0)){
        if((rw4[b]==0)&&(cl2[b]==0)&&(pd4[b]==0)&&(pb5[b]==0)){
          nm[9]=a; nm[17]=b;
          uflg[a]++; rw2[a]=1; cl4[a]=1; pd3[a]=1; pb5[a]=1;
          uflg[b]++; rw4[b]=1; cl2[b]=1; pd4[b]=1; pb5[b]=1;
```

```
            stp12();
            uflg[b]--; rw4[b]=0; cl2[b]=0; pd4[b]=0; pb5[b]=0;
            uflg[a]--; rw2[a]=0; cl4[a]=0; pd3[a]=0; pb5[a]=0;}}}
  }
}
// Set n12{3,2,5,4} and n14{3,4,2,1}
void stp12(){
 short a,b;
  for(a=4;a>=0;a--){b=SC-a;
    if((uflg[a]<5)&&(uflg[b]<5)){
      if((rw3[a]==0)&&(cl2[a]==0)&&(pd5[a]==0)&&(pb4[a]==0)){
        if((rw3[b]==0)&&(cl4[b]==0)&&(pd2[b]==0)&&(pb1[b]==0)){
          nm[12]=a; nm[14]=b;
          uflg[a]++; rw3[a]=1; cl2[a]=1; pd5[a]=1; pb4[a]=1;
          uflg[b]++; rw3[b]=1; cl4[b]=1; pd2[b]=1; pb1[b]=1;
          lurecord();
          uflg[b]--; rw3[b]=0; cl4[b]=0; pd2[b]=0; pb1[b]=0;
          uflg[a]--; rw3[a]=0; cl2[a]=0; pd5[a]=0; pb4[a]=0;}}}
  }
}
//
// Record The Latin Units
void lurecord(){
 short n;
  tlu[cnt][0]=cnt+1;
  for(n=1;n<26;n++){tlu[cnt][n]=nm[n];}
  cnt++;
}
//
// Print Latin Units
void prlunit(short x){
 short t,l,l5,m,n,p;
  for(t=0;t<lcnt;t=t+x){
    p=x; if(t+x>lcnt){p=lcnt-t;}
    for(m=0;m<p;m++){printf("  %9d/",tlu[t+m][0]);}
    printf("\n");
    for(l=0;l<5;l++){l5=l*5;
      for(m=0;m<p;m++){
        printf("   ");
        for(n=1;n<6;n++){printf("%2d",tlu[t+m][l5+n]);}
      }
      printf("\n");
    }
  }
  printf(" [Count of Latin Units = %d]\n",lcnt);
 //* Measure How Similar or How Complementary each pair is *
  for(m=0;m<cnt;m++){
    for(n=0;n<cnt;n++){
      t=0;
      for(l=1;l<26;l++){if(tlu[m][l]==tlu[n][l]){t++;}}
      mtc[m][n]=t;
      t=0;
      for(l=1;l<26;l++){if(tlu[m][l]+tlu[n][l]==4){t++;}}
      if(mtc[m][n]<t){mtc[m][n]=t;}
    }
  }
//
//  . . .(skip). . .
//
//* Combine and Compose *
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<(lcnt/2);u1++){
    for(u2=0;u2<lcnt;u2++){
      if(mtc[u1][u2]==5){
```

```
          for(n=1;n<26;n++){uflg[n]=0;}
          for(n=1;n<26;n++){
            d=tlu[u1][n]*5+tlu[u2][n]+1;
            nm[n]=d; uflg[d]++;
          }
          fc=0;
          for(n=1;n<26;n++){if(uflg[n]==0){fc++;}}
          if(fc==0){
            if((nm[1]<nm[5])&&(nm[5]<nm[21])&&(nm[1]<nm[25])){solsavep();}}
        }
      }
    }
}
//
// Save Answers to the Solution List and Print
void solsavep(){
 short n;
  cnt++;
  anm[0][0]=cnt;
  for(n=1;n<26;n++){anm[0][n]=nm[n];}
  for(n=1;n<26;n++){tn[n]=nm[n]-1; anm[1][n]=tn[n];}; chksms(1);
  for(n=1;n<26;n++){anm[2][n]=tlu[u1][n];}; anm[2][26]=u1;
  for(n=1;n<26;n++){anm[3][n]=tlu[u2][n];}; anm[3][26]=u2;
  pr1ans();
}
//
//  . . .(skip). . .
//
```

** Simultaneous Magic Squares of Order 5: Self-complementary and Pan-diagonal;
'Complete Euler Type' Reconstructed by the 'Greco-Latinian Method' **

** List of Latin Units by /D5i **

| 1/ | 2/ | 3/ | 4/ | 5/ | 6/ | 7/ | 8/ |
|---|---|---|---|---|---|---|---|
| 0 4 3 2 1 | 0 2 4 1 3 | 0 4 1 2 3 | 0 2 4 3 1 | 1 3 4 2 0 | 1 2 3 0 4 | 1 3 0 2 4 | 1 2 3 4 0 |
| 2 1 0 4 3 | 4 1 3 0 2 | 2 3 0 4 1 | 4 3 1 0 2 | 2 0 1 3 4 | 3 0 4 1 2 | 2 4 1 3 0 | 3 4 0 1 2 |
| 4 3 2 1 0 | 3 0 2 4 1 | 4 1 2 3 0 | 1 0 2 4 3 | 3 4 2 0 1 | 4 1 2 3 0 | 3 0 2 4 1 | 0 1 2 3 4 |
| 1 0 4 3 2 | 2 4 1 3 0 | 3 0 4 1 2 | 2 4 3 1 0 | 0 1 3 4 2 | 2 3 0 4 1 | 4 1 3 0 2 | 2 3 4 0 1 |
| 3 2 1 0 4 | 1 3 0 2 4 | 1 2 3 0 4 | 3 1 0 2 4 | 4 2 0 1 3 | 0 4 1 2 3 | 0 2 4 1 3 | 4 0 1 2 3 |

| 9/ | 10/ | 11/ | 12/ | 13/ | 14/ | 15/ | 16/ |
|---|---|---|---|---|---|---|---|
| 3 2 1 0 4 | 3 1 4 2 0 | 3 2 1 4 0 | 3 1 0 2 4 | 4 2 0 1 3 | 4 0 3 2 1 | 4 2 0 3 1 | 4 0 1 2 3 |
| 1 0 4 3 2 | 2 0 3 1 4 | 1 4 0 3 2 | 2 4 3 1 0 | 0 1 3 4 2 | 2 1 4 0 3 | 0 3 1 4 2 | 2 3 4 0 1 |
| 4 3 2 1 0 | 1 4 2 0 3 | 0 3 2 1 4 | 1 0 2 4 3 | 3 4 2 0 1 | 0 3 2 1 4 | 1 4 2 0 3 | 0 1 2 3 4 |
| 2 1 0 4 3 | 0 3 1 4 2 | 2 1 4 0 3 | 4 3 1 0 2 | 2 0 1 3 4 | 1 4 0 3 2 | 2 0 3 1 4 | 3 4 0 1 2 |
| 0 4 3 2 1 | 4 2 0 3 1 | 4 0 3 2 1 | 0 2 4 3 1 | 1 3 4 2 0 | 3 2 1 4 0 | 3 1 4 2 0 | 1 2 3 4 0 |

[Count of Latin Units = 16]

[Reference Table for the Best Combination]

| 0\| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1\| | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 |
| 2\| | 5 | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 5 |
| 3\| | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 |
| 4\| | 5 | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 5 |
| 5\| | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 5 | 5 | 15 | 5 | 25 | 5 | 5 | 5 | 5 |
| 6\| | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 |
| 7\| | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 5 | 5 | 25 | 5 | 15 | 5 | 5 | 5 | 5 |
| 8\| | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 |
| 9\| | 5 | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 25 | 5 | 15 | 5 | 5 | 5 | 5 | 5 |
| 10\| | 5 | 5 | 5 | 5 | 15 | 5 | 25 | 5 | 5 | 25 | 5 | 15 | 5 | 5 | 5 | 5 |
| 11\| | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 |
| 12\| | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 5 | 5 | 15 | 5 | 25 | 5 | 5 | 5 | 5 |
| 13\| | 5 | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 | 5 |
| 14\| | 15 | 5 | 25 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 25 | 5 | 15 |

```
15|  5 25  5 15  5  5  5  5  5  5  5  5 15  5 25  5
16| 25  5 15  5  5  5  5  5  5  5  5  5  5 15  5 25

* Complete Euler Type of Simultaneous MS55: Self-complementary & Pan-diagonal; *
  ** Primitive List of Reconstruction with Full Data: Check Sums **
  [Latin Units: High, Low; [Mathem]; Check Sums:|Row|Column\Pd1/Pd2|; [S.Nmb]]
  /LU    1H          2L   [Math]         |Rw|Cl\P1/P2|   [ 1]
  0 4 3 2 1   0 2 4 1 3    0 22 19 11  8|60|60|60|60|    1 23 20 12  9
  2 1 0 4 3   4 1 3 0 2   14  6  3 20 17|60|60|60|60|   15  7  4 21 18
  4 3 2 1 0   3 0 2 4 1   23 15 12  9  1|60|60|60|60|   24 16 13 10  2
  1 0 4 3 2   2 4 1 3 0    7  4 21 18 10|60|60|60|60|    8  5 22 19 11
  3 2 1 0 4   1 3 0 2 4   16 13  5  2 24|60|60|60|60|   17 14  6  3 25
  /LU    1H          4L   [Math]         |Rw|Cl\P1/P2|   [ 2]
  0 4 3 2 1   0 2 4 3 1    0 22 19 13  6|60|60|60|60|    1 23 20 14  7
  2 1 0 4 3   4 3 1 0 2   14  8  1 20 17|60|60|60|60|   15  9  2 21 18
  4 3 2 1 0   1 0 2 4 3   21 15 12  9  3|60|60|60|60|   22 16 13 10  4
  1 0 4 3 2   2 4 3 1 0    7  4 23 16 10|60|60|60|60|    8  5 24 17 11
  3 2 1 0 4   3 1 0 2 4   18 11  5  2 24|60|60|60|60|   19 12  6  3 25
  /LU    1H          6L   [Math]         |Rw|Cl\P1/P2|   [ 3]
  0 4 3 2 1   1 2 3 0 4    1 22 18 10  9|60|60|60|60|    2 23 19 11 10
  2 1 0 4 3   3 0 4 1 2   13  5  4 21 17|60|60|60|60|   14  6  5 22 18
  4 3 2 1 0   4 1 2 3 0   24 16 12  8  0|60|60|60|60|   25 17 13  9  1
  1 0 4 3 2   2 3 0 4 1    7  3 20 19 11|60|60|60|60|    8  4 21 20 12
  3 2 1 0 4   0 4 1 2 3   15 14  6  2 23|60|60|60|60|   16 15  7  3 24
  /LU    1H          8L   [Math]         |Rw|Cl\P1/P2|   [ 4]
  0 4 3 2 1   1 2 3 4 0    1 22 18 14  5|60|60|60|60|    2 23 19 15  6
  2 1 0 4 3   3 4 0 1 2   13  9  0 21 17|60|60|60|60|   14 10  1 22 18
  4 3 2 1 0   0 1 2 3 4   20 16 12  8  4|60|60|60|60|   21 17 13  9  5
  1 0 4 3 2   2 3 4 0 1    7  3 24 15 11|60|60|60|60|    8  4 25 16 12
  3 2 1 0 4   4 0 1 2 3   19 10  6  2 23|60|60|60|60|   20 11  7  3 24
  /LU    1H          9L   [Math]         |Rw|Cl\P1/P2|   [ 5]
  0 4 3 2 1   3 2 1 0 4    3 22 16 10  9|60|60|60|60|    4 23 17 11 10
  2 1 0 4 3   1 0 4 3 2   11  5  4 23 17|60|60|60|60|   12  6  5 24 18
  4 3 2 1 0   4 3 2 1 0   24 18 12  6  0|60|60|60|60|   25 19 13  7  1
  1 0 4 3 2   2 1 0 4 3    7  1 20 19 13|60|60|60|60|    8  2 21 20 14
  3 2 1 0 4   0 4 3 2 1   15 14  8  2 21|60|60|60|60|   16 15  9  3 22
  /LU    1H         11L   [Math]         |Rw|Cl\P1/P2|   [ 6]
  0 4 3 2 1   3 2 1 4 0    3 22 16 14  5|60|60|60|60|    4 23 17 15  6
  2 1 0 4 3   1 4 0 3 2   11  9  0 23 17|60|60|60|60|   12 10  1 24 18
  4 3 2 1 0   0 3 2 1 4   20 18 12  6  4|60|60|60|60|   21 19 13  7  5
  1 0 4 3 2   2 1 4 0 3    7  1 24 15 13|60|60|60|60|    8  2 25 16 14
  3 2 1 0 4   4 0 3 2 1   19 10  8  2 21|60|60|60|60|   20 11  9  3 22
  /LU    1H         13L   [Math]         |Rw|Cl\P1/P2|   [ 7]
  0 4 3 2 1   4 2 0 1 3    4 22 15 11  8|60|60|60|60|    5 23 16 12  9
  2 1 0 4 3   0 1 3 4 2   10  6  3 24 17|60|60|60|60|   11  7  4 25 18
  4 3 2 1 0   3 4 2 0 1   23 19 12  5  1|60|60|60|60|   24 20 13  6  2
  1 0 4 3 2   2 0 1 3 4    7  0 21 18 14|60|60|60|60|    8  1 22 19 15
  3 2 1 0 4   1 3 4 2 0   16 13  9  2 20|60|60|60|60|   17 14 10  3 21
  /LU    1H         15L   [Math]         |Rw|Cl\P1/P2|   [ 8]
  0 4 3 2 1   4 2 0 3 1    4 22 15 13  6|60|60|60|60|    5 23 16 14  7
  2 1 0 4 3   0 3 1 4 2   10  8  1 24 17|60|60|60|60|   11  9  2 25 18
  4 3 2 1 0   1 4 2 0 3   21 19 12  5  3|60|60|60|60|   22 20 13  6  4
  1 0 4 3 2   2 0 3 1 4    7  0 23 16 14|60|60|60|60|    8  1 24 17 15
  3 2 1 0 4   3 1 4 2 0   18 11  9  2 20|60|60|60|60|   19 12 10  3 21
  /LU    4H          1L   [Math]         |Rw|Cl\P1/P2|   [ 9]
  0 2 4 3 1   0 4 3 2 1    0 14 23 17  6|60|60|60|60|    1 15 24 18  7
```

```
 4 3 1 0 2   2 1 0 4 3   22 16  5  4 13|60|60|60|60|   23 17  6  5 14
 1 0 2 4 3   4 3 2 1 0    9  3 12 21 15|60|60|60|60|   10  4 13 22 16
 2 4 3 1 0   1 0 4 3 2   11 20 19  8  2|60|60|60|60|   12 21 20  9  3
 3 1 0 2 4   3 2 1 0 4   18  7  1 10 24|60|60|60|60|   19  8  2 11 25
 /LU    4H         3L   [Math]        |Rw|Cl\P1/P2|   [10]
 0 2 4 3 1   0 4 1 2 3    0 14 21 17  8|60|60|60|60|    1 15 22 18  9
 4 3 1 0 2   2 3 0 4 1   22 18  5  4 11|60|60|60|60|   23 19  6  5 12
 1 0 2 4 3   4 1 2 3 0    9  1 12 23 15|60|60|60|60|   10  2 13 24 16
 2 4 3 1 0   3 0 4 1 2   13 20 19  6  2|60|60|60|60|   14 21 20  7  3
 3 1 0 2 4   1 2 3 0 4   16  7  3 10 24|60|60|60|60|   17  8  4 11 25
 /LU    4H         5L   [Math]        |Rw|Cl\P1/P2|   [11]
 0 2 4 3 1   1 3 4 2 0    1 13 24 17  5|60|60|60|60|    2 14 25 18  6
 4 3 1 0 2   2 0 1 3 4   22 15  6  3 14|60|60|60|60|   23 16  7  4 15
 1 0 2 4 3   3 4 2 0 1    8  4 12 20 16|60|60|60|60|    9  5 13 21 17
 2 4 3 1 0   0 1 3 4 2   10 21 18  9  2|60|60|60|60|   11 22 19 10  3
 3 1 0 2 4   4 2 0 1 3   19  7  0 11 23|60|60|60|60|   20  8  1 12 24
 /LU    4H         7L   [Math]        |Rw|Cl\P1/P2|   [12]
 0 2 4 3 1   1 3 0 2 4    1 13 20 17  9|60|60|60|60|    2 14 21 18 10
 4 3 1 0 2   2 4 1 3 0   22 19  6  3 10|60|60|60|60|   23 20  7  4 11
 1 0 2 4 3   3 0 2 4 1    8  0 12 24 16|60|60|60|60|    9  1 13 25 17
 2 4 3 1 0   4 1 3 0 2   14 21 18  5  2|60|60|60|60|   15 22 19  6  3
 3 1 0 2 4   0 2 4 1 3   15  7  4 11 23|60|60|60|60|   16  8  5 12 24
 /LU    4H        10L   [Math]        |Rw|Cl\P1/P2|   [13]
 0 2 4 3 1   3 1 4 2 0    3 11 24 17  5|60|60|60|60|    4 12 25 18  6
 4 3 1 0 2   2 0 3 1 4   22 15  8  1 14|60|60|60|60|   23 16  9  2 15
 1 0 2 4 3   1 4 2 0 3    6  4 12 20 18|60|60|60|60|    7  5 13 21 19
 2 4 3 1 0   0 3 1 4 2   10 23 16  9  2|60|60|60|60|   11 24 17 10  3
 3 1 0 2 4   4 2 0 3 1   19  7  0 13 21|60|60|60|60|   20  8  1 14 22
 /LU    4H        12L   [Math]        |Rw|Cl\P1/P2|   [14]
 0 2 4 3 1   3 1 0 2 4    3 11 20 17  9|60|60|60|60|    4 12 21 18 10
 4 3 1 0 2   2 4 3 1 0   22 19  8  1 10|60|60|60|60|   23 20  9  2 11
 1 0 2 4 3   1 0 2 4 3    6  0 12 24 18|60|60|60|60|    7  1 13 25 19
 2 4 3 1 0   4 3 1 0 2   14 23 16  5  2|60|60|60|60|   15 24 17  6  3
 3 1 0 2 4   0 2 4 3 1   15  7  4 13 21|60|60|60|60|   16  8  5 14 22
 /LU    4H        14L   [Math]        |Rw|Cl\P1/P2|   [15]
 0 2 4 3 1   4 0 3 2 1    4 10 23 17  6|60|60|60|60|    5 11 24 18  7
 4 3 1 0 2   2 1 4 0 3   22 16  9  0 13|60|60|60|60|   23 17 10  1 14
 1 0 2 4 3   0 3 2 1 4    5  3 12 21 19|60|60|60|60|    6  4 13 22 20
 2 4 3 1 0   1 4 0 3 2   11 24 15  8  2|60|60|60|60|   12 25 16  9  3
 3 1 0 2 4   3 2 1 4 0   18  7  1 14 20|60|60|60|60|   19  8  2 15 21
 /LU    4H        16L   [Math]        |Rw|Cl\P1/P2|   [16]
 0 2 4 3 1   4 0 1 2 3    4 10 21 17  8|60|60|60|60|    5 11 22 18  9
 4 3 1 0 2   2 3 4 0 1   22 18  9  0 11|60|60|60|60|   23 19 10  1 12
 1 0 2 4 3   0 1 2 3 4    5  1 12 23 19|60|60|60|60|    6  2 13 24 20
 2 4 3 1 0   3 4 0 1 2   13 24 15  6  2|60|60|60|60|   14 25 16  7  3
 3 1 0 2 4   1 2 3 4 0   16  7  3 14 20|60|60|60|60|   17  8  4 15 21
[Solution Counts = 16]  [OK!]
** Calculated and listed by Kanji Setsuda
    on Apr.15, 2012 with MacOSX 10.7.3 & Xcode 4.2.1 **
```

These 16 reconstructions are just the same with the ones we got before by our ordinary method. We could neither make any other type of solutions, nor miss any necessary one. They verified the result of **Knight's Tour Composition**, and proved its validity as a composing method.

How beautiful these 16 **Suzuki Squares** are! Whenever I come to see them, I cannot

help thinking that way. And I am so happy to know that our method clarifies the real 'inner structure' of those precious jewels by the 5th Increment Number System.

## 5. How to build Simultaneous Magic Squares of Order 7: Both S-C & P-D

Can we build Simultaneous MS77: Self-Complementary and Pan-Diagonal by this new method by the 7th Increment System? We have to solve the next problem.

```
** Conceptual Diagrams for New Euler's Method **
 [Decomposition by the 7-th Increment Number System]
 [Classic]         *#   [Math]              /D7i       */H|Rw|Cl\P1/P2|          **/L|Rw|Cl\P1/P2|
  1 28 48 19 39 10 30    0 27 47 18 38  9 29   0 3 6 2 5 1 4|21|21|21|21|  0 6 5 4 3 2 1|21|21|21|21|
 38  9 29  7 27 47 18   37  8 28  6 26 46 17   5 1 4 0 3 6 2|21|21|21|21|  2 1 0 6 5 4 3|21|21|21|21|
 26 46 17 37  8 35  6   25 45 16 36  7 34  5   3 6 2 5 1 4 0|21|21|21|21|  4 3 2 1 0 6 5|21|21|21|21|
 14 34  5 25 45 16 36   13 33  4 24 44 15 35   1 4 0 3 6 2 5|21|21|21|21|  6 5 4 3 2 1 0|21|21|21|21|
 44 15 42 13 33  4 24   43 14 41 12 32  3 23   6 2 5 1 4 0 3|21|21|21|21|  1 0 6 5 4 3 2|21|21|21|21|
 32  3 23 43 21 41 12   31  2 22 42 20 40 11   4 0 3 6 2 5 1|21|21|21|21|  3 2 1 0 6 5 4|21|21|21|21|
 20 40 11 31  2 22 49   19 39 10 30  1 21 48   2 5 1 4 0 3 6|21|21|21|21|  5 4 3 2 1 0 6|21|21|21|21|
 (Calculation: Divide [Math] by 7, and put the integer part of quotient into /D7i High unit
               and put the remainder into Low.)

 [Composition by the 7-th Increment Number System]
 /LU       */H          **/L   [Math]             |Row|Clm\Pd1/Pd2|  [Classic]         *#
  0 3 6 2 5 1 4   0 6 5 4 3 2 1   0 27 47 18 38  9 29|168|168|168|168|   1 28 48 19 39 10 30
  5 1 4 0 3 6 2   2 1 0 6 5 4 3  37  8 28  6 26 46 17|168|168|168|168|  38  9 29  7 27 47 18
  3 6 2 5 1 4 0   4 3 2 1 0 6 5  25 45 16 36  7 34  5|168|168|168|168|  26 46 17 37  8 35  6
  1 4 0 3 6 2 5   6 5 4 3 2 1 0  13 33  4 24 44 15 35|168|168|168|168|  14 34  5 25 45 16 36
  6 2 5 1 4 0 3   1 0 6 5 4 3 2  43 14 41 12 32  3 23|168|168|168|168|  44 15 42 13 33  4 24
  4 0 3 6 2 5 1   3 2 1 0 6 5 4  31  2 22 42 20 40 11|168|168|168|168|  32  3 23 43 21 41 12
  2 5 1 4 0 3 6   5 4 3 2 1 0 6  19 39 10 30  1 21 48|168|168|168|168|  20 40 11 31  2 22 49
 (Calculation: m/H x 7 + n/L = [Math]; [Math] + 1 = [Classic] in the corresponding positions:
               3 x 7 + 6 = 27; 27 + 1 = 28;    5 x 7 + 2 = 37; 37 + 1 = 38; ...
```

Let's compose the **Complete Euler Squares** by the **Greco-Latinian Method** now.

First of all, we have to build all the Latin Units required, and then pick up two of them to combine and compose our whole object by our arithmetic calculation.

I prepared the following basic form and basic conditions for our Latin Units.

```
//
//** Basic Form for the Latin Units of Order 7 **
//                  .--.--.--.--.--.--.
//  2  3  4  5  6  7| 1| 2| 3| 4| 5| 6| 7| 1  2  3  4  5  6
//                  |--+--+--+--+--+--+--|
//  9 10 11 12 13 14| 8| 9|10|11|12|13|14| 8  9 10 11 12 13
//                  |--+--+--+--+--+--+--|
// 16 17 18 19 20 21|15|16|17|18|19|20|21|15 16 17 18 19 20
//                  |--+--+--+--+--+--+--|
// 23 24 25 26 27 28|22|23|24|25|26|27|28|22 23 24 25 26 27
//                  |--+--+--+--+--+--+--|
// 30 31 32 33 34 35|29|30|31|32|33|34|35|29 30 31 32 33 34
//                  |--+--+--+--+--+--+--|
// 37 38 39 40 41 42|36|37|38|39|40|41|42|36 37 38 39 40 41
//                  |--+--+--+--+--+--+--|
// 44 45 46 47 48 49|43|44|45|46|47|48|49|43 44 45 46 47 48
//                  '--'--'--'--'--'--'--'
//** Self-complementary Conditions of the Object: SC=6 **
//  n1+n49=n2+n48=n3+n47=n4+n46=n5+n45=n6+n44=n7+n43=n8+n42=n9+n41=
// n10+n40=n11+n39=n12+n38=n13+n37=n14+n36=n15+n35=n16+n34=n17+n33=n18+n32=
// n19+n31=n20+n30=n21+n29=n22+n28=n23+n27=n24+n26=n25+n25=SC ... scc
//** Definitions of Rows and Columns: C=21 **
//  n1+n2+n3+n4+n5+n6+n7=C        ... rw1  | n1+n8+n15+n22+n29+n36+n43=C   ... cl1
//  n8+n9+n10+n11+n12+n13+n14=C   ... rw2  | n2+n9+n16+n23+n30+n37+n44=C   ... cl2
// n15+n16+n17+n18+n19+n20+n21=C  ... rw3  | n3+n10+n17+n24+n31+n38+n45=C  ... cl3
// n22+n23+n24+n25+n26+n27+n28=C  ... rw4  | n4+n11+n18+n25+n32+n39+n46=C  ... cl4
// n29+n30+n31+n32+n33+n34+n35=C  ... rw5  | n5+n12+n19+n26+n33+n40+n47=C  ... cl5
// n36+n37+n38+n39+n40+n41+n42=C  ... rw6  | n6+n13+n20+n27+n34+n41+n48=C  ... cl6
```

```
// n43+n44+n45+n46+n47+n48+n49=C  ... rw7  | n7+n14+n21+n28+n35+n42+n49=C  ... cl7
//** Pan-diagonal Conditions of the Object: C=21 **
//  n1+n9+n17+n25+n33+n41+n49=C   ... pd1  | n1+n14+n20+n26+n32+n38+n44=C  ... pd8
//  n2+n10+n18+n26+n34+n42+n43=C  ... pd2  | n2+n8+n21+n27+n33+n39+n45=C   ... pd9
//  n3+n11+n19+n27+n35+n36+n44=C  ... pd3  | n3+n9+n15+n28+n34+n40+n46=C   ... pd10
//  n4+n12+n20+n28+n29+n37+n45=C  ... pd4  | n4+n10+n16+n22+n35+n41+n47=C  ... pd11
//  n5+n13+n21+n22+n30+n38+n46=C  ... pd5  | n5+n11+n17+n23+n29+n42+n48=C  ... pd12
//  n6+n14+n15+n23+n31+n39+n47=C  ... pd6  | n6+n12+n18+n24+n30+n36+n49=C  ... pd13
//  n7+n8+n16+n24+n32+n40+n48=C   ... pd7  | n7+n13+n19+n25+n31+n37+n43=C  ... pd14
//** List-forming Inequality Conditions for the Standard Solutions **
//  n1<n43; n43<n7; and n1<n49;
//
```

We have to prepare 28x7 flag counters to check if any number of {0, 1, 2, 3, 4, 5, 6} is used strictly once in each row, column and pan-diagonal. If they follow the Latin Condition, every sum of them is consequently equal to the constant sum 21.

We also have to count how often each number is used for each unit, and check whether it is used **seven times** or not. More or less often it must not be used.

The next list shows the computer program I prepared, only the core part of it.

If you want to have the complete list of program, refer to the Appendix, please.

```
//
#include <stdio.h>
//
short int lcnt, cnt, cnt1, cnt3;
short SC, LSM;
short u1, u2;
short nm[50], uflg[50];
short dn[5][52];
short tlu[193][50];
short mtc[193][193];
short st[3457][52];
short tn[50];
short cs[4][29];
//
short rw1[7], cl1[7], pd1[7], pb1[7];
short rw2[7], cl2[7], pd2[7], pb2[7];
short rw3[7], cl3[7], pd3[7], pb3[7];
short rw4[7], cl4[7], pd4[7], pb4[7];
short rw5[7], cl5[7], pd5[7], pb5[7];
short rw6[7], cl6[7], pd6[7], pb6[7];
short rw7[7], cl7[7], pd7[7], pb7[7];
//
//  . . .(skip). . .
//
//* Main Program *
int main(){
 short n;
  printf("\n");
  printf("** 'Complete Euler Type' of Simultaneous Magic Squares of Order 7: **\n");
  printf("** Self-complementary and Pan-diagonal; made by New Euler's Method **\n");
  for(n=0;n<50;n++){nm[n]=0;}
  for(n=0;n<7;n++){uflg[n]=0;
     rw1[n]=0; cl1[n]=0; pd1[n]=0; pb1[n]=0;
     rw2[n]=0; cl2[n]=0; pd2[n]=0; pb2[n]=0;
     rw3[n]=0; cl3[n]=0; pd3[n]=0; pb3[n]=0;
     rw4[n]=0; cl4[n]=0; pd4[n]=0; pb4[n]=0;
     rw5[n]=0; cl5[n]=0; pd5[n]=0; pb5[n]=0;
     rw6[n]=0; cl6[n]=0; pd6[n]=0; pb6[n]=0;
     rw7[n]=0; cl7[n]=0; pd7[n]=0; pb7[n]=0;
  }
  SC=6; LSM=21; cnt=0; cnt3=0;
  stp00();                      // Make the Latin Units
  slusort();                    // Sort the Latin Units
  lcnt=cnt;
```

```
  refmatch();                   // Make the Reference Table of Matching
  cnt=0;
  printf("\n");
  printf(" [List of Latin Units]\n");
  prlunit();
  LSM=175; cnt=0; cnt1=0; cnt3=0;
  printf("\n");
  printf("** Complete Euler Type of Simultaneous MS77: Self-Complementary and Pan-diagonal; **\n");
  printf(" [Latin Units: /High /Low; Reconstruction: Sol_Number#; Sum_Errors:|Row|Clm\\Pd1/Pd2|]\n");
  cmbcmp();                     // Combine and Compose
  solsort();                    // Sort the Solutions
  solprint();                   // Print the Solution List
  if(cnt3>0){prsol(cnt3);}      // Print The Rest One
  printf(" [Total Counts of Solutions([n1=1] All#) = [%d] %d#]   [OK!]\n",cnt1,cnt);
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on Apr.13, 2012 with MacOSX 10.7.3 and Xcode 4.3.2 **\n");
  printf("\n");
 return 0;
}
//
//* Make the Latin Units by the 7th Increment System
// Level #0:
// Set n25{4,4,1,7}
void stp00(){
 short a;
  a=3; nm[25]=a;
  uflg[a]++; rw4[a]=1; cl4[a]=1; pd1[a]=1; pb7[a]=1;
  stp01();
  uflg[a]--; rw4[a]=0; cl4[a]=0; pd1[a]=0; pb7[a]=0;
}
// Level #1:
// Set n1 & n49=SC-N1
void stp01(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl1[a]==0)&&(pd1[a]==0)&&(pb1[a]==0)){
        if((rw7[b]==0)&&(cl7[b]==0)&&(pd1[b]==0)&&(pb6[b]==0)){
          nm[1]=a; nm[49]=b;
          uflg[a]++; rw1[a]=1; cl1[a]=1; pd1[a]=1; pb1[a]=1;
          uflg[b]++; rw7[b]=1; cl7[b]=1; pd1[b]=1; pb6[b]=1;
          stp02();
          uflg[b]--; rw7[b]=0; cl7[b]=0; pd1[b]=0; pb6[b]=0;
          uflg[a]--; rw1[a]=0; cl1[a]=0; pd1[a]=0; pb1[a]=0;}}}
  }
}
// Set n9 & n41=SC-N9
void stp02(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw2[a]==0)&&(cl2[a]==0)&&(pd1[a]==0)&&(pb3[a]==0)){
        if((rw6[b]==0)&&(cl6[b]==0)&&(pd1[b]==0)&&(pb4[b]==0)){
          nm[9]=a; nm[41]=b;
          uflg[a]++; rw2[a]=1; cl2[a]=1; pd1[a]=1; pb3[a]=1;
          uflg[b]++; rw6[b]=1; cl6[b]=1; pd1[b]=1; pb4[b]=1;
          stp03();
          uflg[b]--; rw6[b]=0; cl6[b]=0; pd1[b]=0; pb4[b]=0;
          uflg[a]--; rw2[a]=0; cl2[a]=0; pd1[a]=0; pb3[a]=0;}}}
  }
}
// Set n17 & n33=SC-N17
void stp03(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
```

```
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw3[a]==0)&&(cl3[a]==0)&&(pd1[a]==0)&&(pb5[a]==0)){
        if((rw5[b]==0)&&(cl5[b]==0)&&(pd1[b]==0)&&(pb2[b]==0)){
          nm[17]=a; nm[33]=b;
          uflg[a]++; rw3[a]=1; cl3[a]=1; pd1[a]=1; pb5[a]=1;
          uflg[b]++; rw5[b]=1; cl5[b]=1; pd1[b]=1; pb2[b]=1;
          stp04();
          uflg[b]--; rw5[b]=0; cl5[b]=0; pd1[b]=0; pb2[b]=0;
          uflg[a]--; rw3[a]=0; cl3[a]=0; pd1[a]=0; pb5[a]=0;}}}
  }
}
// Level #2:
// Set n2 & n48=SC-N2
void stp04(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl2[a]==0)&&(pd2[a]==0)&&(pb2[a]==0)){
        if((rw7[b]==0)&&(cl6[b]==0)&&(pd7[b]==0)&&(pb5[b]==0)){
          nm[2]=a; nm[48]=b;
          uflg[a]++; rw1[a]=1; cl2[a]=1; pd2[a]=1; pb2[a]=1;
          uflg[b]++; rw7[b]=1; cl6[b]=1; pd7[b]=1; pb5[b]=1;
          stp05();
          uflg[b]--; rw7[b]=0; cl6[b]=0; pd7[b]=0; pb5[b]=0;
          uflg[a]--; rw1[a]=0; cl2[a]=0; pd2[a]=0; pb2[a]=0;}}}
  }
}
// Set n3 & n47=SC-N3
void stp05(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl3[a]==0)&&(pd3[a]==0)&&(pb3[a]==0)){
        if((rw7[b]==0)&&(cl5[b]==0)&&(pd6[b]==0)&&(pb4[b]==0)){
          nm[3]=a; nm[47]=b;
          uflg[a]++; rw1[a]=1; cl3[a]=1; pd3[a]=1; pb3[a]=1;
          uflg[b]++; rw7[b]=1; cl5[b]=1; pd6[b]=1; pb4[b]=1;
          stp06();
          uflg[b]--; rw7[b]=0; cl5[b]=0; pd6[b]=0; pb4[b]=0;
          uflg[a]--; rw1[a]=0; cl3[a]=0; pd3[a]=0; pb3[a]=0;}}}
  }
}
// Set n4 & n46=SC-N4
void stp06(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl4[a]==0)&&(pd4[a]==0)&&(pb4[a]==0)){
        if((rw7[b]==0)&&(cl4[b]==0)&&(pd5[b]==0)&&(pb3[b]==0)){
          nm[4]=a; nm[46]=b;
          uflg[a]++; rw1[a]=1; cl4[a]=1; pd4[a]=1; pb4[a]=1;
          uflg[b]++; rw7[b]=1; cl4[b]=1; pd5[b]=1; pb3[b]=1;
          stp07();
          uflg[b]--; rw7[b]=0; cl4[b]=0; pd5[b]=0; pb3[b]=0;
          uflg[a]--; rw1[a]=0; cl4[a]=0; pd4[a]=0; pb4[a]=0;}}}
  }
}
// Set n5 & n45=SC-N5
void stp07(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl5[a]==0)&&(pd5[a]==0)&&(pb5[a]==0)){
        if((rw7[b]==0)&&(cl3[b]==0)&&(pd4[b]==0)&&(pb2[b]==0)){
          nm[5]=a; nm[45]=b;
```

```
            uflg[a]++; rw1[a]=1; cl5[a]=1; pd5[a]=1; pb5[a]=1;
            uflg[b]++; rw7[b]=1; cl3[b]=1; pd4[b]=1; pb2[b]=1;
            stp08();
            uflg[b]--; rw7[b]=0; cl3[b]=0; pd4[b]=0; pb2[b]=0;
            uflg[a]--; rw1[a]=0; cl5[a]=0; pd5[a]=0; pb5[a]=0;}}}
  }
}
// Set n6 & n44=SC-N6
void stp08(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl6[a]==0)&&(pd6[a]==0)&&(pb6[a]==0)){
        if((rw7[b]==0)&&(cl2[b]==0)&&(pd3[b]==0)&&(pb1[b]==0)){
          nm[6]=a; nm[44]=b;
          uflg[a]++; rw1[a]=1; cl6[a]=1; pd6[a]=1; pb6[a]=1;
          uflg[b]++; rw7[b]=1; cl2[b]=1; pd3[b]=1; pb1[b]=1;
          stp09();
          uflg[b]--; rw7[b]=0; cl2[b]=0; pd3[b]=0; pb1[b]=0;
          uflg[a]--; rw1[a]=0; cl6[a]=0; pd6[a]=0; pb6[a]=0;}}}
  }
}
// Set n7 & n43=SC-N7
void stp09(){
 short a,b;
  for(a=6;a>=0;a--){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw1[a]==0)&&(cl7[a]==0)&&(pd7[a]==0)&&(pb7[a]==0)){
        if((rw7[b]==0)&&(cl1[b]==0)&&(pd2[b]==0)&&(pb7[b]==0)){
          nm[7]=a; nm[43]=b;
          uflg[a]++; rw1[a]=1; cl7[a]=1; pd7[a]=1; pb7[a]=1;
          uflg[b]++; rw7[b]=1; cl1[b]=1; pd2[b]=1; pb7[b]=1;
          stp10();
          uflg[b]--; rw7[b]=0; cl1[b]=0; pd2[b]=0; pb7[b]=0;
          uflg[a]--; rw1[a]=0; cl7[a]=0; pd7[a]=0; pb7[a]=0;}}}
  }
}
// Level #3:
// Set n8 & n42=SC-N8
void stp10(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw2[a]==0)&&(cl1[a]==0)&&(pd7[a]==0)&&(pb2[a]==0)){
        if((rw6[b]==0)&&(cl7[b]==0)&&(pd2[b]==0)&&(pb5[b]==0)){
          nm[8]=a; nm[42]=b;
          uflg[a]++; rw2[a]=1; cl1[a]=1; pd7[a]=1; pb2[a]=1;
          uflg[b]++; rw6[b]=1; cl7[b]=1; pd2[b]=1; pb5[b]=1;
          stp11();
          uflg[b]--; rw6[b]=0; cl7[b]=0; pd2[b]=0; pb5[b]=0;
          uflg[a]--; rw2[a]=0; cl1[a]=0; pd7[a]=0; pb2[a]=0;}}}
  }
}
// Set n15 & n35=SC-N15
void stp11(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw3[a]==0)&&(cl1[a]==0)&&(pd6[a]==0)&&(pb3[a]==0)){
        if((rw5[b]==0)&&(cl7[b]==0)&&(pd3[b]==0)&&(pb4[b]==0)){
          nm[15]=a; nm[35]=b;
          uflg[a]++; rw3[a]=1; cl1[a]=1; pd6[a]=1; pb3[a]=1;
          uflg[b]++; rw5[b]=1; cl7[b]=1; pd3[b]=1; pb4[b]=1;
          stp12();
          uflg[b]--; rw5[b]=0; cl7[b]=0; pd3[b]=0; pb4[b]=0;
```

```
            uflg[a]--; rw3[a]=0; cl1[a]=0; pd6[a]=0; pb3[a]=0;}}}
  }
}
// Set n22 & n28=SC-N22
void stp12(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw4[a]==0)&&(cl1[a]==0)&&(pd5[a]==0)&&(pb4[a]==0)){
        if((rw4[b]==0)&&(cl7[b]==0)&&(pd4[b]==0)&&(pb3[b]==0)){
          nm[22]=a; nm[28]=b;
          uflg[a]++; rw4[a]=1; cl1[a]=1; pd5[a]=1; pb4[a]=1;
          uflg[b]++; rw4[b]=1; cl7[b]=1; pd4[b]=1; pb3[b]=1;
          stp13();
          uflg[b]--; rw4[b]=0; cl7[b]=0; pd4[b]=0; pb3[b]=0;
          uflg[a]--; rw4[a]=0; cl1[a]=0; pd5[a]=0; pb4[a]=0;}}}
  }
}
// Set n29 & n21=SC-N29
void stp13(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw5[a]==0)&&(cl1[a]==0)&&(pd4[a]==0)&&(pb5[a]==0)){
        if((rw3[b]==0)&&(cl7[b]==0)&&(pd5[b]==0)&&(pb2[b]==0)){
          nm[29]=a; nm[21]=b;
          uflg[a]++; rw5[a]=1; cl1[a]=1; pd4[a]=1; pb5[a]=1;
          uflg[b]++; rw3[b]=1; cl7[b]=1; pd5[b]=1; pb2[b]=1;
          stp14();
          uflg[b]--; rw3[b]=0; cl7[b]=0; pd5[b]=0; pb2[b]=0;
          uflg[a]--; rw5[a]=0; cl1[a]=0; pd4[a]=0; pb5[a]=0;}}}
  }
}
// Set n36 & n14=SC-N36
void stp14(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw6[a]==0)&&(cl1[a]==0)&&(pd3[a]==0)&&(pb6[a]==0)){
        if((rw2[b]==0)&&(cl7[b]==0)&&(pd6[b]==0)&&(pb1[b]==0)){
          nm[36]=a; nm[14]=b;
          uflg[a]++; rw6[a]=1; cl1[a]=1; pd3[a]=1; pb6[a]=1;
          uflg[b]++; rw2[b]=1; cl7[b]=1; pd6[b]=1; pb1[b]=1;
          stp15();
          uflg[b]--; rw2[b]=0; cl7[b]=0; pd6[b]=0; pb1[b]=0;
          uflg[a]--; rw6[a]=0; cl1[a]=0; pd3[a]=0; pb6[a]=0;}}}
  }
}
//
// Level #4:
//  . . .(skip). . .
//  . . .(skip). . .
// Set n18 & n32=SC-N18
void stp23(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw3[a]==0)&&(cl4[a]==0)&&(pd2[a]==0)&&(pb6[a]==0)){
        if((rw5[b]==0)&&(cl4[b]==0)&&(pd7[b]==0)&&(pb1[b]==0)){
          nm[18]=a; nm[32]=b;
          uflg[a]++; rw3[a]=1; cl4[a]=1; pd2[a]=1; pb6[a]=1;
          uflg[b]++; rw5[b]=1; cl4[b]=1; pd7[b]=1; pb1[b]=1;
          stp24();
          uflg[b]--; rw5[b]=0; cl4[b]=0; pd7[b]=0; pb1[b]=0;
          uflg[a]--; rw3[a]=0; cl4[a]=0; pd2[a]=0; pb6[a]=0;}}}
```

```
  }
}
// Set n24 & n26=SC-N24
void stp24(){
 short a,b;
  for(a=0;a<7;a++){b=SC-a;
    if((uflg[a]<7)&&(uflg[b]<7)){
      if((rw4[a]==0)&&(cl3[a]==0)&&(pd7[a]==0)&&(pb6[a]==0)){
        if((rw4[b]==0)&&(cl5[b]==0)&&(pd2[b]==0)&&(pb1[b]==0)){
          nm[24]=a; nm[26]=b;
          uflg[a]++; rw4[a]=1; cl3[a]=1; pd7[a]=1; pb6[a]=1;
          uflg[b]++; rw4[b]=1; cl5[b]=1; pd2[b]=1; pb1[b]=1;
          lurecord();
          uflg[b]--; rw4[b]=0; cl5[b]=0; pd2[b]=0; pb1[b]=0;
          uflg[a]--; rw4[a]=0; cl3[a]=0; pd7[a]=0; pb6[a]=0;}}}
  }
}
//
//* Record Each of the Latin Units *
void lurecord(){
 short n;
  tlu[cnt][0]=cnt+1;
  for(n=1;n<50;n++){tlu[cnt][n]=nm[n];}
  cnt++;
}
//
//  . . .(skip). . .
//
//* Make the Reference Table of Matching *
void refmatch(){
 short t,l,m,n;
  for(m=0;m<lcnt;m++){
    for(n=0;n<lcnt;n++){
      t=0;
      for(l=1;l<50;l++){if(tlu[m][l]==tlu[n][l]){t++;}}
      mtc[m][n]=t;
      t=0;
      for(l=1;l<50;l++){if(tlu[m][l]+tlu[n][l]==6){t++;}}
      if(mtc[m][n]<t){mtc[m][n]=t;}
    }
  }
}
//
//* Print the Latin Units *
void prlunit(){
 short t,l,l7,m,n;
  for(t=0;t<lcnt;t=t+48){
    printf("%8d/%8d/%8d/%8d/%8d/%8d/%8d/%8d/\n",
           t+1,t+2,t+3,t+4,t+5,t+6,t+7,t+8);
    for(l=0;l<7;l++){l7=l*7;
      for(m=t;m<(t+8);m++){
        printf("  ");
        for(n=1;n<8;n++){printf("%d",tlu[m][l7+n]);}
      }
      printf("\n");
    }
  }
  printf(" [Count of Latin Units = %d]\n",lcnt);
//
//  . . .(skip). . .
//
//* Combine and Compose *
void cmbcmp(){
 short n,d,fc;
  for(u1=0;u1<(lcnt/2);u1++){
```

```
    for(u2=0;u2<lcnt;u2++){
      if(mtc[u1][u2]==7){
        for(n=1;n<50;n++){uflg[n]=0;}
        for(n=1;n<50;n++){
          d=tlu[u1][n]*7+tlu[u2][n]+1;
          nm[n]=d; uflg[d]++;
        }
        fc=0;
        for(n=1;n<50;n++){if(uflg[n]==1){fc++;}else{break;}}
        if(fc==49){
          if((nm[1]<nm[49])&&(nm[1]<nm[43])&&(nm[43]<nm[7])){recsol();}
        }
      }
    }
  }
}
//
//* Record Each of the Solutions *
void recsol(){
 short n;
  cnt++;
  st[cnt][0]=cnt;
  for(n=1;n<50;n++){st[cnt][n]=nm[n];}
  st[cnt][50]=u1+1; st[cnt][51]=u2+1;
  if(nm[1]==1){cnt1++;}
}
//
//  . . .(skip). . .
//
```

The next lists show the execution results of this program in the abstract style.

```
** 'Complete Euler Type' of Simultaneous Magic Squares of Order 7: **
** Self-complementary and Pan-diagonal; made by New Euler's Method **

 [List of Latin Units]
      1/      2/      3/      4/      5/      6/      7/      8/
 0654321 0531642 0362514 0246135 0652341 0531624 0426135 0364512
 2106543 3164205 5140362 6135024 4106523 3162405 6135042 5120364
 4321065 6420531 3625140 5024613 2341065 6240531 5042613 3645120
 6543210 2053164 1403625 4613502 6523410 4053162 2613504 1203645
 1065432 5316420 6251403 3502461 1065234 5316240 3504261 6451203
 3210654 1642053 4036251 2461350 3410652 1624053 4261350 2036451
 5432106 4205316 2514036 1350246 5234106 2405316 1350426 4512036
     49/     50/     51/     52/     53/     54/     55/     56/
 1526304 1350246 1234560 1065432 1605432 1520364 1356240 1234506
 0415263 2461350 3456012 5432106 5432160 6415203 2401356 3450612
 6304152 3502461 5601234 2106543 2160543 0364152 3562401 5061234
 5263041 4613502 0123456 6543210 0543216 5203641 4013562 6123450
 4152630 5024613 2345601 3210654 3216054 4152036 5624013 2345061
 3041526 6135024 4560123 0654321 6054321 3641520 0135624 4506123
 2630415 0246135 6012345 4321065 4321605 2036415 6240135 0612345
     97/     98/     99/    100/    101/    102/    103/    104/
 4630251 4321605 4265310 4152036 4630215 4512036 4325601 4261350
 3025146 6054321 1042653 2036415 3021546 2036451 6014325 5042613
 2514630 3216054 5310426 6415203 2154630 6451203 3256014 1350426
 1463025 0543216 2653104 5203641 5463021 1203645 0143256 2613504
 6302514 2160543 0426531 3641520 6302154 3645120 2560143 0426135
 0251463 5432160 3104265 1520364 0215463 5120364 1432560 3504261
 5146302 1605432 6531042 0364152 1546302 0364512 5601432 6135042
    145/    146/    147/    148/    149/    150/    151/    152/
 5310246 5234160 5126304 5061432 5601432 5316240 5234106 5120364
 2465310 3416052 0451263 1432506 1432560 2405316 3410652 6451203
```

```
 3102465  1605234  6304512  2506143  2560143  3162405  1065234  0364512
 4653102  0523416  1263045  6143250  0143256  4053162  6523410  1203645
 1024653  2341605  4512630  3250614  3256014  1624053  2341065  4512036
 6531024  4160523  3045126  0614325  6014325  0531624  4106523  3645120
 0246531  6052341  2630451  4325061  4325601  6240531  0652341  2036451
[Count of Latin Units = 192]

[Reference Table for the Best Combination]
 0|  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 ...
--+-------------------------------------------------------------------------
 1| 49  7  7  7 35  7  7  7 21  7  7  7 21  7  7  7 21  7  7  7 21  7  7  7 ...
 2|  7 49  7  7  7 35  7  7  7 21  7  7  7  7 21  7  7  7 21  7  7  7  7 21
 3|  7  7 49  7  7  7  7 35  7  7 21  7  7  7  7 21  7 21  7  7  7  7 21  7
 4|  7  7  7 49  7  7 35  7  7  7  7 21  7 21  7  7  7  7  7 21  7 21  7  7
 5| 35  7  7  7 49  7  7  7 21  7  7  7 21  7  7  7 21  7  7  7 21  7  7  7
 6|  7 35  7  7  7 49  7  7  7 21  7  7  7  7 21  7  7  7 21  7  7  7  7 21
 7|  7  7  7 35  7  7 49  7  7  7  7 21  7 21  7  7  7  7  7 21  7 21  7  7
 8|  7  7 35  7  7  7  7 49  7  7 21  7  7  7  7 21  7 21  7  7  7  7 21  7
 9| 21  7  7  7 21  7  7  7 49  7  7  7 35  7  7  7 35  7  7  7 35  7  7  7
10|  7 21  7  7  7 21  7  7  7 49  7  7  7  7 35  7  7  7 35  7  7  7  7 35
11|  7  7 21  7  7  7 21  7  7 49  7  7  7  7 35  7 35  7  7  7  7 35  7
12|  7  7  7 21  7  7 21  7  7  7  7 49  7 35  7  7  7  7  7 35  7 35  7  7
13| 21  7  7  7 21  7  7  7 35  7  7  7 49  7  7  7 35  7  7  7 35  7  7  7
14|  7  7  7 21  7  7 21  7  7  7  7 35  7 49  7  7  7  7  7 35  7 35  7  7
15|  7 21  7  7  7 21  7  7  7 35  7  7  7  7 49  7  7  7 35  7  7  7  7 35
16|  7  7 21  7  7  7  7 21  7  7 35  7  7  7  7 49  7 35  7  7  7  7 35  7
17| 21  7  7  7 21  7  7  7 35  7  7  7 35  7  7  7 49  7  7  7 35  7  7  7
18|  7  7 21  7  7  7  7 21  7  7 35  7  7  7  7 35  7 49  7  7  7  7 35  7
19|  7 21  7  7  7 21  7  7  7 35  7  7  7  7 35  7  7  7 49  7  7  7  7 35
20|  7  7  7 21  7  7 21  7  7  7  7 35  7 35  7  7  7  7  7 49  7 35  7  7
21| 21  7  7  7 21  7  7  7 35  7  7  7 35  7  7  7 35  7  7  7 49  7  7  7
22|  7  7  7 21  7  7 21  7  7  7  7 35  7 35  7  7  7  7  7 35  7 49  7  7
23|  7  7 21  7  7  7  7 21  7  7 35  7  7  7  7 35  7 35  7  7  7  7 49  7
24|  7 21  7  7  7 21  7  7  7 35  7  7  7  7 35  7  7  7 35  7  7  7  7 49
...| ...

** Complete Euler Type of Simultaneous MS77: Self-Complementary and Pan-diagonal; **
 [Latin Units: /High /Low; Reconstruction: Sol_Number#; Sum_Errors:|Row|Clm\Pd1/Pd2|]
    3/H     1/L                    1#|R|C\P/P|      3/H     2/L                    2#|R|C\P/P|
0362514 0654321   1 28 48 19 39 10 30|0|0|0|0|   0362514 0531642   1 27 46 16 42 12 31|0|0|0|0|
5140362 2106543  38  9 29  7 27 47 18|0|0|0|0|   5140362 3164205  39  9 35  5 24 43 20|0|0|0|0|
3625140 4321065  26 46 17 37  8 35  6|0|0|0|0|   3625140 6420531  28 47 17 36 13 32  2|0|0|0|0|
1403625 6543210  14 34  5 25 45 16 36|0|0|0|0|   1403625 2053164  10 29  6 25 44 21 40|0|0|0|0|
6251403 1065432  44 15 42 13 33  4 24|0|0|0|0|   6251403 5316420  48 18 37 14 33  3 22|0|0|0|0|
4036251 3210654  32  3 23 43 21 41 12|0|0|0|0|   4036251 1642053  30  7 26 45 15 41 11|0|0|0|0|
2514036 5432106  20 40 11 31  2 22 49|0|0|0|0|   2514036 4205316  19 38  8 34  4 23 49|0|0|0|0|

    3/H     4/L                    3#|R|C\P/P|      4/H     1/L                    4#|R|C\P/P|
0362514 0246135   1 24 47 21 37 11 34|0|0|0|0|   0246135 0654321   1 21 34 47 11 24 37|0|0|0|0|
5140362 6135024  42  9 32  6 22 45 19|0|0|0|0|   6135024 2106543  45  9 22 42  6 19 32|0|0|0|0|
3625140 5024613  27 43 17 40 14 30  4|0|0|0|0|   5024613 4321065  40  4 17 30 43 14 27|0|0|0|0|
1403625 4613502  12 35  2 25 48 15 38|0|0|0|0|   4613502 6543210  35 48 12 25 38  2 15|0|0|0|0|
6251403 3502461  46 20 36 10 33  7 23|0|0|0|0|   3502461 1065432  23 36  7 20 33 46 10|0|0|0|0|
4036251 2461350  31  5 28 44 18 41  8|0|0|0|0|   2461350 3210654  18 31 44  8 28 41  5|0|0|0|0|
2514036 1350246  16 39 13 29  3 26 49|0|0|0|0|   1350246 5432106  13 26 39  3 16 29 49|0|0|0|0|

    4/H     2/L                    5#|R|C\P/P|      4/H     3/L                    6#|R|C\P/P|
0246135 0531642   1 20 32 44 14 26 38|0|0|0|0|   0246135 0362514   1 18 35 45 13 23 40|0|0|0|0|
6135024 3164205  46  9 28 40  3 15 34|0|0|0|0|   6135024 5140362  48  9 26 36  4 21 31|0|0|0|0|
5024613 6420531  42  5 17 29 48 11 23|0|0|0|0|   5024613 3625140  39  7 17 34 44 12 22|0|0|0|0|
4613502 2053164  31 43 13 25 37  7 19|0|0|0|0|   4613502 1403625  30 47  8 25 42  3 20|0|0|0|0|
3502461 5316420  27 39  2 21 33 45  8|0|0|0|0|   3502461 6251403  28 38  6 16 33 43 11|0|0|0|0|
2461350 1642053  16 35 47 10 22 41  4|0|0|0|0|   2461350 4036251  19 29 46 14 24 41  2|0|0|0|0|
1350246 4205316  12 24 36  6 18 30 49|0|0|0|0|   1350246 2514036  10 27 37  5 15 32 49|0|0|0|0|

    3/H     5/L                    7#|R|C\P/P|      3/H     6/L                    8#|R|C\P/P|
0362514 0652341   1 28 48 17 39 12 30|0|0|0|0|   0362514 0531624   1 27 46 16 42 10 33|0|0|0|0|
5140362 4106523  40  9 29  7 27 45 18|0|0|0|0|   5140362 3162405  39  9 35  3 26 43 20|0|0|0|0|
```

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3625140 | 2341065 | 24 | 46 | 19 | 37 | 8 | 35 | 6 | 0 | 0 | 0 | 0 | | |
| 1403625 | 6523410 | 14 | 34 | 3 | 25 | 47 | 16 | 36 | 0 | 0 | 0 | 0 | | |
| 6251403 | 1065234 | 44 | 15 | 42 | 13 | 31 | 4 | 26 | 0 | 0 | 0 | 0 | | |
| 4036251 | 3410652 | 32 | 5 | 23 | 43 | 21 | 41 | 10 | 0 | 0 | 0 | 0 | | |
| 2514036 | 5234106 | 20 | 38 | 11 | 33 | 2 | 22 | 49 | 0 | 0 | 0 | 0 | | |

| 3/H | 7/L | | | | | | | 9# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0362514 | 0426135 | 1 | 26 | 45 | 21 | 37 | 11 | 34 | 0 | 0 | 0 | 0 | | |
| 5140362 | 6135042 | 42 | 9 | 32 | 6 | 22 | 47 | 17 | 0 | 0 | 0 | 0 | | |
| 3625140 | 5042613 | 27 | 43 | 19 | 38 | 14 | 30 | 4 | 0 | 0 | 0 | 0 | | |
| 1403625 | 2613504 | 10 | 35 | 2 | 25 | 48 | 15 | 40 | 0 | 0 | 0 | 0 | | |
| 6251403 | 3504261 | 46 | 20 | 36 | 12 | 31 | 7 | 23 | 0 | 0 | 0 | 0 | | |
| 4036251 | 4261350 | 33 | 3 | 28 | 44 | 18 | 41 | 8 | 0 | 0 | 0 | 0 | | |
| 2514036 | 1350426 | 16 | 39 | 13 | 29 | 5 | 24 | 49 | 0 | 0 | 0 | 0 | | |

| 4/H | 6/L | | | | | | | 11# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0246135 | 0531624 | 1 | 20 | 32 | 44 | 14 | 24 | 40 | 0 | 0 | 0 | 0 | | |
| 6135024 | 3162405 | 46 | 9 | 28 | 38 | 5 | 15 | 34 | 0 | 0 | 0 | 0 | | |
| 5024613 | 6240531 | 42 | 3 | 19 | 29 | 48 | 11 | 23 | 0 | 0 | 0 | 0 | | |
| 4613502 | 4053162 | 33 | 43 | 13 | 25 | 37 | 7 | 17 | 0 | 0 | 0 | 0 | | |
| 3502461 | 5316240 | 27 | 39 | 2 | 21 | 31 | 47 | 8 | 0 | 0 | 0 | 0 | | |
| 2461350 | 1624053 | 16 | 35 | 45 | 12 | 22 | 41 | 4 | 0 | 0 | 0 | 0 | | |
| 1350246 | 2405316 | 10 | 26 | 36 | 6 | 18 | 30 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 1/L | | | | | | | 13# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0654321 | 1 | 42 | 27 | 12 | 46 | 17 | 30 | 0 | 0 | 0 | 0 | | |
| 3162405 | 2106543 | 24 | 9 | 43 | 21 | 34 | 5 | 39 | 0 | 0 | 0 | 0 | | |
| 6240531 | 4321065 | 47 | 18 | 31 | 2 | 36 | 28 | 13 | 0 | 0 | 0 | 0 | | |
| 4053162 | 6543210 | 35 | 6 | 40 | 25 | 10 | 44 | 15 | 0 | 0 | 0 | 0 | | |
| 5316240 | 1065432 | 37 | 22 | 14 | 48 | 19 | 32 | 3 | 0 | 0 | 0 | 0 | | |
| 1624053 | 3210654 | 11 | 45 | 16 | 29 | 7 | 41 | 26 | 0 | 0 | 0 | 0 | | |
| 2405316 | 5432106 | 20 | 33 | 4 | 38 | 23 | 8 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 4/L | | | | | | | 15# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0246135 | 1 | 38 | 26 | 14 | 44 | 18 | 34 | 0 | 0 | 0 | 0 | | |
| 3162405 | 6135024 | 28 | 9 | 46 | 20 | 29 | 3 | 40 | 0 | 0 | 0 | 0 | | |
| 6240531 | 5024613 | 48 | 15 | 31 | 5 | 42 | 23 | 11 | 0 | 0 | 0 | 0 | | |
| 4053162 | 4613502 | 33 | 7 | 37 | 25 | 13 | 43 | 17 | 0 | 0 | 0 | 0 | | |
| 5316240 | 3502461 | 39 | 27 | 8 | 45 | 19 | 35 | 2 | 0 | 0 | 0 | 0 | | |
| 1624053 | 2461350 | 10 | 47 | 21 | 30 | 4 | 41 | 22 | 0 | 0 | 0 | 0 | | |
| 2405316 | 1350246 | 16 | 32 | 6 | 36 | 24 | 12 | 49 | 0 | 0 | 0 | 0 | | |

| 7/H | 2/L | | | | | | | 17# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0531642 | 1 | 34 | 18 | 44 | 14 | 26 | 38 | 0 | 0 | 0 | 0 | | |
| 6135042 | 3164205 | 46 | 9 | 28 | 40 | 3 | 29 | 20 | 0 | 0 | 0 | 0 | | |
| 5042613 | 6420531 | 42 | 5 | 31 | 15 | 48 | 11 | 23 | 0 | 0 | 0 | 0 | | |
| 2613504 | 2053164 | 17 | 43 | 13 | 25 | 37 | 7 | 33 | 0 | 0 | 0 | 0 | | |
| 3504261 | 5316420 | 27 | 39 | 2 | 35 | 19 | 45 | 8 | 0 | 0 | 0 | 0 | | |
| 4261350 | 1642053 | 30 | 21 | 47 | 10 | 22 | 41 | 4 | 0 | 0 | 0 | 0 | | |
| 1350426 | 4205316 | 12 | 24 | 36 | 6 | 32 | 16 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 5/L | | | | | | | 19# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0652341 | 1 | 42 | 27 | 10 | 46 | 19 | 30 | 0 | 0 | 0 | 0 | | |
| 3162405 | 4106523 | 26 | 9 | 43 | 21 | 34 | 3 | 39 | 0 | 0 | 0 | 0 | | |
| 6240531 | 2341065 | 45 | 18 | 33 | 2 | 36 | 28 | 13 | 0 | 0 | 0 | 0 | | |
| 4053162 | 6523410 | 35 | 6 | 38 | 25 | 12 | 44 | 15 | 0 | 0 | 0 | 0 | | |
| 5316240 | 1065234 | 37 | 22 | 14 | 48 | 17 | 32 | 5 | 0 | 0 | 0 | 0 | | |
| 1624053 | 3410652 | 11 | 47 | 16 | 29 | 7 | 41 | 24 | 0 | 0 | 0 | 0 | | |
| 2405316 | 5234106 | 20 | 31 | 4 | 40 | 23 | 8 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 8/L | | | | | | | 21# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0364512 | 1 | 39 | 28 | 12 | 48 | 16 | 31 | 0 | 0 | 0 | 0 | | |
| 3162405 | 5120364 | 27 | 9 | 45 | 15 | 32 | 7 | 40 | 0 | 0 | 0 | 0 | | |
| 6240531 | 3645120 | 46 | 21 | 33 | 6 | 37 | 24 | 8 | 0 | 0 | 0 | 0 | | |
| 4053162 | 1203645 | 30 | 3 | 36 | 25 | 14 | 47 | 20 | 0 | 0 | 0 | 0 | | |
| 5316240 | 6451203 | 42 | 26 | 13 | 44 | 17 | 29 | 4 | 0 | 0 | 0 | 0 | | |
| 1624053 | 2036451 | 10 | 43 | 18 | 35 | 5 | 41 | 23 | 0 | 0 | 0 | 0 | | |
| 2405316 | 4512036 | 19 | 34 | 2 | 38 | 22 | 11 | 49 | 0 | 0 | 0 | 0 | | |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3625140 | 6240531 | 28 | 45 | 19 | 36 | 13 | 32 | 2 | 0 | 0 | 0 | 0 | | |
| 1403625 | 4053162 | 12 | 29 | 6 | 25 | 44 | 21 | 38 | 0 | 0 | 0 | 0 | | |
| 6251403 | 5316240 | 48 | 18 | 37 | 14 | 31 | 5 | 22 | 0 | 0 | 0 | 0 | | |
| 4036251 | 1624053 | 30 | 7 | 24 | 47 | 15 | 41 | 11 | 0 | 0 | 0 | 0 | | |
| 2514036 | 2405316 | 17 | 40 | 8 | 34 | 4 | 23 | 49 | 0 | 0 | 0 | 0 | | |

| 4/H | 5/L | | | | | | | 10# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0246135 | 0652341 | 1 | 21 | 34 | 45 | 11 | 26 | 37 | 0 | 0 | 0 | 0 | | |
| 6135024 | 4106523 | 47 | 9 | 22 | 42 | 6 | 17 | 32 | 0 | 0 | 0 | 0 | | |
| 5024613 | 2341065 | 38 | 4 | 19 | 30 | 43 | 14 | 27 | 0 | 0 | 0 | 0 | | |
| 4613502 | 6523410 | 35 | 48 | 10 | 25 | 40 | 2 | 15 | 0 | 0 | 0 | 0 | | |
| 3502461 | 1065234 | 23 | 36 | 7 | 20 | 31 | 46 | 12 | 0 | 0 | 0 | 0 | | |
| 2461350 | 3410652 | 18 | 33 | 44 | 8 | 28 | 41 | 3 | 0 | 0 | 0 | 0 | | |
| 1350246 | 5234106 | 13 | 24 | 39 | 5 | 16 | 29 | 49 | 0 | 0 | 0 | 0 | | |

| 4/H | 8/L | | | | | | | 12# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0246135 | 0364512 | 1 | 18 | 35 | 47 | 13 | 23 | 38 | 0 | 0 | 0 | 0 | | |
| 6135024 | 5120364 | 48 | 9 | 24 | 36 | 4 | 21 | 33 | 0 | 0 | 0 | 0 | | |
| 5024613 | 3645120 | 39 | 7 | 19 | 34 | 44 | 10 | 22 | 0 | 0 | 0 | 0 | | |
| 4613502 | 1203645 | 30 | 45 | 8 | 25 | 42 | 5 | 20 | 0 | 0 | 0 | 0 | | |
| 3502461 | 6451203 | 28 | 40 | 6 | 16 | 31 | 43 | 11 | 0 | 0 | 0 | 0 | | |
| 2461350 | 2036451 | 17 | 29 | 46 | 14 | 26 | 41 | 2 | 0 | 0 | 0 | 0 | | |
| 1350246 | 4512036 | 12 | 27 | 37 | 3 | 15 | 32 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 3/L | | | | | | | 14# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0362514 | 1 | 39 | 28 | 10 | 48 | 16 | 33 | 0 | 0 | 0 | 0 | | |
| 3162405 | 5140362 | 27 | 9 | 47 | 15 | 32 | 7 | 38 | 0 | 0 | 0 | 0 | | |
| 6240531 | 3625140 | 46 | 21 | 31 | 6 | 37 | 26 | 8 | 0 | 0 | 0 | 0 | | |
| 4053162 | 1403625 | 30 | 5 | 36 | 25 | 14 | 45 | 20 | 0 | 0 | 0 | 0 | | |
| 5316240 | 6251403 | 42 | 24 | 13 | 44 | 19 | 29 | 4 | 0 | 0 | 0 | 0 | | |
| 1624053 | 4036251 | 12 | 43 | 18 | 35 | 3 | 41 | 23 | 0 | 0 | 0 | 0 | | |
| 2405316 | 2514036 | 17 | 34 | 2 | 40 | 22 | 11 | 49 | 0 | 0 | 0 | 0 | | |

| 7/H | 1/L | | | | | | | 16# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0654321 | 1 | 35 | 20 | 47 | 11 | 24 | 37 | 0 | 0 | 0 | 0 | | |
| 6135042 | 2106543 | 45 | 9 | 22 | 42 | 6 | 33 | 18 | 0 | 0 | 0 | 0 | | |
| 5042613 | 4321065 | 40 | 4 | 31 | 16 | 43 | 14 | 27 | 0 | 0 | 0 | 0 | | |
| 2613504 | 6543210 | 21 | 48 | 12 | 25 | 38 | 2 | 29 | 0 | 0 | 0 | 0 | | |
| 3504261 | 1065432 | 23 | 36 | 7 | 34 | 19 | 46 | 10 | 0 | 0 | 0 | 0 | | |
| 4261350 | 3210654 | 32 | 17 | 44 | 8 | 28 | 41 | 5 | 0 | 0 | 0 | 0 | | |
| 1350426 | 5432106 | 13 | 26 | 39 | 3 | 30 | 15 | 49 | 0 | 0 | 0 | 0 | | |

| 7/H | 3/L | | | | | | | 18# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0362514 | 1 | 32 | 21 | 45 | 13 | 23 | 40 | 0 | 0 | 0 | 0 | | |
| 6135042 | 5140362 | 48 | 9 | 26 | 36 | 4 | 35 | 17 | 0 | 0 | 0 | 0 | | |
| 5042613 | 3625140 | 39 | 7 | 31 | 20 | 44 | 12 | 22 | 0 | 0 | 0 | 0 | | |
| 2613504 | 1403625 | 16 | 47 | 8 | 25 | 42 | 3 | 34 | 0 | 0 | 0 | 0 | | |
| 3504261 | 6251403 | 28 | 38 | 6 | 30 | 19 | 43 | 11 | 0 | 0 | 0 | 0 | | |
| 4261350 | 4036251 | 33 | 15 | 46 | 14 | 24 | 41 | 2 | 0 | 0 | 0 | 0 | | |
| 1350426 | 2514036 | 10 | 27 | 37 | 5 | 29 | 18 | 49 | 0 | 0 | 0 | 0 | | |

| 6/H | 7/L | | | | | | | 20# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0531624 | 0426135 | 1 | 40 | 24 | 14 | 44 | 18 | 34 | 0 | 0 | 0 | 0 | | |
| 3162405 | 6135042 | 28 | 9 | 46 | 20 | 29 | 5 | 38 | 0 | 0 | 0 | 0 | | |
| 6240531 | 5042613 | 48 | 15 | 33 | 3 | 42 | 23 | 11 | 0 | 0 | 0 | 0 | | |
| 4053162 | 2613504 | 31 | 7 | 37 | 25 | 13 | 43 | 19 | 0 | 0 | 0 | 0 | | |
| 5316240 | 3504261 | 39 | 27 | 8 | 47 | 17 | 35 | 2 | 0 | 0 | 0 | 0 | | |
| 1624053 | 4261350 | 12 | 45 | 21 | 30 | 4 | 41 | 22 | 0 | 0 | 0 | 0 | | |
| 2405316 | 1350426 | 16 | 32 | 6 | 36 | 26 | 10 | 49 | 0 | 0 | 0 | 0 | | |

| 7/H | 5/L | | | | | | | 22# | R | C\P/P | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0652341 | 1 | 35 | 20 | 45 | 11 | 26 | 37 | 0 | 0 | 0 | 0 | | |
| 6135042 | 4106523 | 47 | 9 | 22 | 42 | 6 | 31 | 18 | 0 | 0 | 0 | 0 | | |
| 5042613 | 2341065 | 38 | 4 | 33 | 16 | 43 | 14 | 27 | 0 | 0 | 0 | 0 | | |
| 2613504 | 6523410 | 21 | 48 | 10 | 25 | 40 | 2 | 29 | 0 | 0 | 0 | 0 | | |
| 3504261 | 1065234 | 23 | 36 | 7 | 34 | 17 | 46 | 12 | 0 | 0 | 0 | 0 | | |
| 4261350 | 3410652 | 32 | 19 | 44 | 8 | 28 | 41 | 3 | 0 | 0 | 0 | 0 | | |
| 1350426 | 5234106 | 13 | 24 | 39 | 5 | 30 | 15 | 49 | 0 | 0 | 0 | 0 | | |

| 7/H | 6/L | | | | | | | 23# | R | C\P/P | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0531624 | 1 | 34 | 18 | 44 | 14 | 24 | 40 | 0 | 0 | 0 | 0 |
| 6135042 | 3162405 | 46 | 9 | 28 | 38 | 5 | 29 | 20 | 0 | 0 | 0 | 0 |
| 5042613 | 6240531 | 42 | 3 | 33 | 15 | 48 | 11 | 23 | 0 | 0 | 0 | 0 |
| 2613504 | 4053162 | 19 | 43 | 13 | 25 | 37 | 7 | 31 | 0 | 0 | 0 | 0 |
| 3504261 | 5316240 | 27 | 39 | 2 | 35 | 17 | 47 | 8 | 0 | 0 | 0 | 0 |
| 4261350 | 1624053 | 30 | 21 | 45 | 12 | 22 | 41 | 4 | 0 | 0 | 0 | 0 |
| 1350426 | 2405316 | 10 | 26 | 36 | 6 | 32 | 16 | 49 | 0 | 0 | 0 | 0 |

| 7/H | 8/L | | | | | | | 24# | R | C\P/P | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0426135 | 0364512 | 1 | 32 | 21 | 47 | 13 | 23 | 38 | 0 | 0 | 0 | 0 |
| 6135042 | 5120364 | 48 | 9 | 24 | 36 | 4 | 35 | 19 | 0 | 0 | 0 | 0 |
| 5042613 | 3645120 | 39 | 7 | 33 | 20 | 44 | 10 | 22 | 0 | 0 | 0 | 0 |
| 2613504 | 1203645 | 16 | 45 | 8 | 25 | 42 | 5 | 34 | 0 | 0 | 0 | 0 |
| 3504261 | 6451203 | 28 | 40 | 6 | 30 | 17 | 43 | 11 | 0 | 0 | 0 | 0 |
| 4261350 | 2036451 | 31 | 15 | 46 | 14 | 26 | 41 | 2 | 0 | 0 | 0 | 0 |
| 1350426 | 4512036 | 12 | 27 | 37 | 3 | 29 | 18 | 49 | 0 | 0 | 0 | 0 |

| 3/H | 9/L | 25# | | | | | 3/H | 17/L | 49# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 28 | 47 | 20 | 39 | 9 | 31 | 1 | 28 | 45 | 20 | 39 | 9 | 33 |
| 37 | 10 | 29 | 7 | 26 | 48 | 18 | 37 | 12 | 29 | 7 | 24 | 48 | 18 |
| 27 | 46 | 16 | 38 | 8 | 35 | 5 | 27 | 46 | 16 | 40 | 8 | 35 | 3 |
| 14 | 33 | 6 | 25 | 44 | 17 | 36 | 14 | 31 | 6 | 25 | 44 | 19 | 36 |
| 45 | 15 | 42 | 12 | 34 | 4 | 23 | 47 | 15 | 42 | 10 | 34 | 4 | 23 |
| 32 | 2 | 24 | 43 | 21 | 40 | 13 | 32 | 2 | 26 | 43 | 21 | 38 | 13 |
| 19 | 41 | 11 | 30 | 3 | 22 | 49 | 17 | 41 | 11 | 30 | 5 | 22 | 49 |

| 3/H | 25/L | 73# | | | | | 11/H | 1/L | 97# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 28 | 44 | 19 | 39 | 10 | 34 | 1 | 28 | 48 | 12 | 32 | 17 | 37 |
| 38 | 13 | 29 | 7 | 23 | 47 | 18 | 31 | 16 | 36 | 7 | 27 | 47 | 11 |
| 26 | 46 | 17 | 41 | 8 | 35 | 2 | 26 | 46 | 10 | 30 | 15 | 42 | 6 |
| 14 | 30 | 5 | 25 | 45 | 20 | 36 | 21 | 41 | 5 | 25 | 45 | 9 | 29 |
| 48 | 15 | 42 | 9 | 33 | 4 | 24 | 44 | 8 | 35 | 20 | 40 | 4 | 24 |
| 32 | 3 | 27 | 43 | 21 | 37 | 12 | 39 | 3 | 23 | 43 | 14 | 34 | 19 |
| 16 | 40 | 11 | 31 | 6 | 22 | 49 | 13 | 33 | 18 | 38 | 2 | 22 | 49 |

| 11/H | 9/L | 121# | | | | | 11/H | 17/L | 145# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 28 | 47 | 13 | 32 | 16 | 38 | 1 | 28 | 45 | 13 | 32 | 16 | 40 |
| 30 | 17 | 36 | 7 | 26 | 48 | 11 | 30 | 19 | 36 | 7 | 24 | 48 | 11 |
| 27 | 46 | 9 | 31 | 15 | 42 | 5 | 27 | 46 | 9 | 33 | 15 | 42 | 3 |
| 21 | 40 | 6 | 25 | 44 | 10 | 29 | 21 | 38 | 6 | 25 | 44 | 12 | 29 |
| 45 | 8 | 35 | 19 | 41 | 4 | 23 | 47 | 8 | 35 | 17 | 41 | 4 | 23 |
| 39 | 2 | 24 | 43 | 14 | 33 | 20 | 39 | 2 | 26 | 43 | 14 | 31 | 20 |
| 12 | 34 | 18 | 37 | 3 | 22 | 49 | 10 | 34 | 18 | 37 | 5 | 22 | 49 |

| 11/H | 25/L | 169# | | | | | 17/H | 2/L | 193# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 28 | 44 | 12 | 32 | 17 | 41 | 1 | 48 | 18 | 37 | 28 | 12 | 31 |
| 31 | 20 | 36 | 7 | 23 | 47 | 11 | 11 | 30 | 7 | 47 | 17 | 36 | 27 |
| 26 | 46 | 10 | 34 | 15 | 42 | 2 | 42 | 26 | 10 | 29 | 6 | 46 | 16 |
| 21 | 37 | 5 | 25 | 45 | 13 | 29 | 45 | 15 | 41 | 25 | 9 | 35 | 5 |
| 48 | 8 | 35 | 16 | 40 | 4 | 24 | 34 | 4 | 44 | 21 | 40 | 24 | 8 |
| 39 | 3 | 27 | 43 | 14 | 30 | 19 | 23 | 14 | 33 | 3 | 43 | 20 | 39 |
| 9 | 33 | 18 | 38 | 6 | 22 | 49 | 19 | 38 | 22 | 13 | 32 | 2 | 49 |

| 17/H | 10/L | 217# | | | | | 17/H | 18/L | 241# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 47 | 18 | 38 | 28 | 13 | 30 | 1 | 46 | 21 | 37 | 24 | 12 | 34 |
| 11 | 31 | 7 | 48 | 16 | 36 | 26 | 10 | 33 | 6 | 43 | 18 | 42 | 23 |
| 42 | 27 | 9 | 29 | 5 | 46 | 17 | 39 | 28 | 9 | 31 | 5 | 48 | 15 |
| 44 | 15 | 40 | 25 | 10 | 35 | 6 | 47 | 20 | 36 | 25 | 14 | 30 | 3 |
| 33 | 4 | 45 | 21 | 41 | 23 | 8 | 35 | 2 | 45 | 19 | 41 | 22 | 11 |
| 24 | 14 | 34 | 2 | 43 | 19 | 39 | 27 | 8 | 32 | 7 | 44 | 17 | 40 |
| 20 | 37 | 22 | 12 | 32 | 3 | 49 | 16 | 38 | 26 | 13 | 29 | 4 | 49 |

| 17/H | 26/L | 265# | | | | | 25/H | 2/L | 289# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 46 | 21 | 38 | 23 | 13 | 33 | 1 | 48 | 11 | 30 | 28 | 19 | 38 |
| 9 | 34 | 5 | 43 | 18 | 42 | 24 | 18 | 37 | 7 | 47 | 10 | 29 | 27 |
| 39 | 28 | 10 | 30 | 6 | 47 | 15 | 35 | 26 | 17 | 36 | 6 | 46 | 9 |
| 48 | 19 | 36 | 25 | 14 | 31 | 2 | 45 | 8 | 34 | 25 | 16 | 42 | 5 |
| 35 | 3 | 44 | 20 | 40 | 22 | 11 | 41 | 4 | 44 | 14 | 33 | 24 | 15 |
| 26 | 8 | 32 | 7 | 45 | 16 | 41 | 23 | 21 | 40 | 3 | 43 | 13 | 32 |
| 17 | 37 | 27 | 12 | 29 | 4 | 49 | 12 | 31 | 22 | 20 | 39 | 2 | 49 |

| 25/H | 10/L | 313# | | | | | 25/H | 18/L | 337# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 47 | 11 | 31 | 28 | 20 | 37 | 1 | 46 | 14 | 30 | 24 | 19 | 41 |
| 18 | 38 | 7 | 48 | 9 | 29 | 26 | 17 | 40 | 6 | 43 | 11 | 35 | 23 |
| 35 | 27 | 16 | 36 | 5 | 46 | 10 | 32 | 28 | 16 | 38 | 5 | 48 | 8 |
| 44 | 8 | 33 | 25 | 17 | 42 | 6 | 47 | 13 | 29 | 25 | 21 | 37 | 3 |
| 40 | 4 | 45 | 14 | 34 | 23 | 15 | 42 | 2 | 45 | 12 | 34 | 22 | 18 |
| 24 | 21 | 41 | 2 | 43 | 12 | 32 | 27 | 15 | 39 | 7 | 44 | 10 | 33 |
| 13 | 30 | 22 | 19 | 39 | 3 | 49 | 9 | 31 | 26 | 20 | 36 | 4 | 49 |

| 25/H | 26/L | 361# | | | | | 3/H | 33/L | 385# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 46 | 14 | 31 | 23 | 20 | 40 | 2 | 28 | 46 | 15 | 41 | 12 | 31 |
| 16 | 41 | 5 | 43 | 11 | 35 | 24 | 39 | 8 | 34 | 5 | 24 | 44 | 21 |
| 32 | 28 | 17 | 37 | 6 | 47 | 8 | 27 | 47 | 17 | 37 | 14 | 32 | 1 |
| 48 | 12 | 29 | 25 | 21 | 38 | 2 | 10 | 30 | 7 | 25 | 43 | 20 | 40 |
| 42 | 3 | 44 | 13 | 33 | 22 | 18 | 49 | 18 | 36 | 13 | 33 | 3 | 23 |
| 26 | 15 | 39 | 7 | 45 | 9 | 34 | 29 | 6 | 26 | 45 | 16 | 42 | 11 |
| 10 | 30 | 27 | 19 | 36 | 4 | 49 | 19 | 38 | 9 | 35 | 4 | 22 | 48 |

| 3/H | 65/L | 769# | | | | | 3/H | 97/L | 1153# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 28 | 46 | 15 | 40 | 13 | 30 | 5 | 28 | 46 | 15 | 38 | 13 | 30 |
| 39 | 8 | 33 | 6 | 23 | 45 | 21 | 39 | 8 | 31 | 6 | 23 | 47 | 21 |
| 26 | 48 | 16 | 38 | 14 | 32 | 1 | 24 | 48 | 16 | 40 | 14 | 32 | 1 |
| 9 | 31 | 7 | 25 | 43 | 19 | 41 | 9 | 33 | 7 | 25 | 43 | 17 | 41 |
| 49 | 18 | 36 | 12 | 34 | 2 | 24 | 49 | 18 | 36 | 10 | 34 | 2 | 26 |
| 29 | 5 | 27 | 44 | 17 | 42 | 11 | 29 | 3 | 27 | 44 | 19 | 42 | 11 |
| 20 | 37 | 10 | 35 | 4 | 22 | 47 | 20 | 37 | 12 | 35 | 4 | 22 | 45 |

| 3/H | 129/L | 1537# | | | | | 3/H | 161/L | 1921# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 28 | 46 | 15 | 37 | 12 | 31 | 7 | 27 | 46 | 16 | 36 | 12 | 31 |
| 39 | 8 | 30 | 5 | 24 | 48 | 21 | 39 | 9 | 29 | 5 | 24 | 49 | 20 |
| 23 | 47 | 17 | 41 | 14 | 32 | 1 | 22 | 47 | 17 | 42 | 13 | 32 | 2 |
| 10 | 34 | 7 | 25 | 43 | 16 | 40 | 10 | 35 | 6 | 25 | 44 | 15 | 40 |
| 49 | 18 | 36 | 9 | 33 | 3 | 27 | 48 | 18 | 37 | 8 | 33 | 3 | 28 |
| 29 | 2 | 26 | 45 | 20 | 42 | 11 | 30 | 1 | 26 | 45 | 21 | 41 | 11 |
| 19 | 38 | 13 | 35 | 4 | 22 | 44 | 19 | 38 | 14 | 34 | 4 | 23 | 43 |

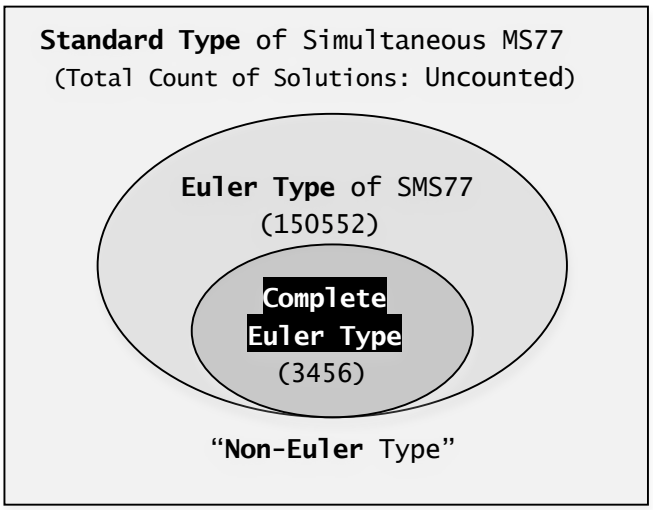
| 35/H | 1/L | 2305# | | | | | 35/H | 33/L | 2497# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 28 | 41 | 19 | 46 | 3 | 30 | 9 | 28 | 39 | 15 | 48 | 5 | 31 |
| 45 | 2 | 29 | 14 | 27 | 40 | 18 | 46 | 1 | 34 | 12 | 24 | 37 | 21 |
| 26 | 39 | 17 | 44 | 1 | 35 | 13 | 27 | 40 | 17 | 44 | 7 | 32 | 8 |
| 7 | 34 | 12 | 25 | 38 | 16 | 43 | 3 | 30 | 14 | 25 | 36 | 20 | 47 |
| 37 | 15 | 49 | 6 | 33 | 11 | 24 | 42 | 18 | 43 | 6 | 33 | 10 | 23 |
| 32 | 10 | 23 | 36 | 21 | 48 | 5 | 29 | 13 | 26 | 38 | 16 | 49 | 4 |
| 20 | 47 | 4 | 31 | 9 | 22 | 42 | 19 | 45 | 2 | 35 | 11 | 22 | 41 |

| 35/H | 65/L | 2689# | | | | | 35/H | 97/L | 2881# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 10 | 28 | 39 | 15 | 47 | 6 | 30 | 12 | 28 | 39 | 15 | 45 | 6 | 30 |
| 46 | 1 | 33 | 13 | 23 | 38 | 21 | 46 | 1 | 31 | 13 | 23 | 40 | 21 |
| 26 | 41 | 16 | 45 | 7 | 32 | 8 | 24 | 41 | 16 | 47 | 7 | 32 | 8 |
| 2 | 31 | 14 | 25 | 36 | 19 | 48 | 2 | 33 | 14 | 25 | 36 | 17 | 48 |
| 42 | 18 | 43 | 5 | 34 | 9 | 24 | 42 | 18 | 43 | 3 | 34 | 9 | 26 |
| 29 | 12 | 27 | 37 | 17 | 49 | 4 | 29 | 10 | 27 | 37 | 19 | 49 | 4 |
| 20 | 44 | 3 | 35 | 11 | 22 | 40 | 20 | 44 | 5 | 35 | 11 | 22 | 38 |

| 35/H | 129/L | 3073# | | | | | 35/H | 161/L | 3265# | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 28 | 39 | 15 | 44 | 5 | 31 | 14 | 27 | 39 | 16 | 43 | 5 | 31 |
| 46 | 1 | 30 | 12 | 24 | 41 | 21 | 46 | 2 | 29 | 12 | 24 | 42 | 20 |
| 23 | 40 | 17 | 48 | 7 | 32 | 8 | 22 | 40 | 17 | 49 | 6 | 32 | 9 |
| 3 | 34 | 14 | 25 | 36 | 16 | 47 | 3 | 35 | 13 | 25 | 37 | 15 | 47 |

```
 42 18 43  2 33 10 27   41 18 44  1 33 10 28
 29  9 26 38 20 49  4   30  8 26 38 21 48  4
 19 45  6 35 11 22 37   19 45  7 34 11 23 36
[Total Counts of Solutions([n1=1] All#) = [384] 3456#]   [OK!]
** Calculated and Listed by Kanji Setsuda
    on Apr.13, 2012 with MacOSX 10.7.3 and Xcode 4.3.2 **
```

These 3456 solutions reconstructed are just the same with the ones we got before.
We could neither make any other type of solutions nor miss any necessary one.
It means we could have verified the validity of our **Knight's Tour Composition**.
And how fast our **Greco-Latinian Method** could run!




In 2012 I knew we could build another set of **Euler Squares** for the Simultaneous Magic Squares 7x7: Self-complementary & Pan-diagonal; with the 150552 standard solutions.

Each **Euler Unit** of them has such the 'similar structure' to its whole object that each row, column and pan-diagonal in it certainly has the equal sum to the constant value 21, though its content pattern does not always follow the Latin Condition.

I want to call "**Euler Squares**" of order 7 for this type, and make the difference from the "**Complete Euler Squares**" made by **Greco-Latinian Method**.

I have reported about it in the Section 11-3, Chapter 4. Please, read it, will you?

## 6. How to build Pan-diagonal Magic Squares of Order 7 by New Euler's Method

I must report that I could also reconstruct 38102400 solutions of Pan-diagonal MS77 by this method, but this article gets a little too long now. I would like to skip my exact report here.

If you want to read that, please click at the last line below to get the program sheet for that type and abstract list of results.

(English Version Written by **Kanji Setsuda** in 2003, 2005 and 2011;
Revised on Apr.21, 2012 with MacOSX 10.7.3 and Xcode 4.3.2)

---

E-Mail Address: jag12001@nifty.com
http://homepage2.nifty.com/KanjiSetsuda/

(Appendix: Program Sheets in full size: Please click here to get them.)