Part 4: "New Advanced Study of Magic Squares and Cubes"
Chapter 4: Commentary Articles No.2 by Kanji Setsuda: Positional Writing System of Numbers: Base N
Section 8: How to build 'Complete Euler Squares' 8x8 by Binary Number System [Revised]

0. Let's build 'Complete Euler Squares' of Order 8.

Why don't we reconstruct various Pan-diagonal Magic Squares of Order 8 by the '**New Euler's Method**' applied even to every pan-diagonal of the object? Now that we can not make any 'Greco-Latinian Square' of Order 8 by Positional Number System of Base 8, instead of giving it up, let's build them by Base 2, that means '**Binary Number System**.'

1. Definition of 'Complete Euler Squares' of Order 8 by Binary Number System

What do they look like by Binary Number System?

Let me show you the next 2 lists that explain the symmetrical processes of Binary Decomposition and Composition of object by the Binary Number System.

Our **New Euler's Method** depends upon these two way methods of decomposition and composition of object by Binary Number System.

** **Decomposition** of 'Composite & Complete' Pan-magic Squares of Order 8 **

[Classical, Mathematical; Decomposed Diagrams:/D2i; Check Sums|RowClmPd1Pd2|]

```
[Clsc]                1#    [Math]                1#          [Serial Divisions]
 1 63  5 59 14 52 10 56      0 62  4 58 13 51  9 55     2) 62          /D2i   2) 50          /D2i
62  4 58  8 49 15 53 11     61  3 57  7 48 14 52 10     2) 31 ... 0 -> /6     2) 25 ... 0 -> /6
17 47 21 43 30 36 26 40     16 46 20 42 29 35 25 39     2) 15 ... 1 -> /5     2) 12 ... 1 -> /5
46 20 42 24 33 31 37 27     45 19 41 23 32 30 36 26     2)  7 ... 1 -> /4     2)  6 ... 0 -> /4
51 13 55  9 64  2 60  6     50 12 54  8 63  1 59  5     2)  3 ... 1 -> /3     2)  3 ... 0 -> /3
16 50 12 54  3 61  7 57     15 49 11 53  2 60  6 56     2)  1 ... 1 -> /2     2)  1 ... 1 -> /2
35 29 39 25 48 18 44 22     34 28 38 24 47 17 43 21        0 ... 1 -> /1          0 ... 1 -> /1
32 34 28 38 19 45 23 41     31 33 27 37 18 44 22 40
```

/D2i	/1|RCPP|	/2|RCPP|	/3|RCPP|	/4|RCPP|	/5|RCPP|	/6|RCPP|
01010101	4444	01010101	4444	01011010	4444	01101001	4444	01010101	4444	00001111	4444
10101010	4444	10101010	4444	10100101	4444	10010110	4444	01010101	4444	11110000	4444
01010101	4444	10101010	4444	01011010	4444	01101001	4444	01010101	4444	00001111	4444
10101010	4444	01010101	4444	10100101	4444	10010110	4444	01010101	4444	11110000	4444
10101010	4444	10101010	4444	01011010	4444	01101001	4444	10101010	4444	00001111	4444
01010101	4444	01010101	4444	10100101	4444	10010110	4444	10101010	4444	11110000	4444
10101010	4444	01010101	4444	01011010	4444	01101001	4444	10101010	4444	00001111	4444
01010101	4444	10101010	4444	10100101	4444	10010110	4444	10101010	4444	11110000	4444

** **Composition** of 'Composite & Complete' Pan-magic Squares of Order 8
by the Binary Number System and the New Euler's Method **

[Euler_Units////// Solution[Classical]: Sol_No#|Sum_Checks||\/|]

[EU] 1/	4/	5/	8/	10/	14/	[Classic]	1#|Row|Clm|Pd1/Pd2|
01010101	01010101	01011010	01101001	01010101	00001111	1 63 5 59 14 52 10 56	260|260|260/260|
10101010	10101010	10100101	10010110	01010101	11110000	62 4 58 8 49 15 53 11	260|260|260/260|
01010101	10101010	01011010	01101001	01010101	00001111	17 47 21 43 30 36 26 40	260|260|260/260|
10101010	01010101	10100101	10010110	01010101	11110000	46 20 42 24 33 31 37 27	260|260|260/260|
10101010	10101010	01011010	01101001	10101010	00001111	51 13 55 9 64 2 60 6	260|260|260/260|
01010101	01010101	10100101	10010110	10101010	11110000	16 50 12 54 3 61 7 57	260|260|260/260|
10101010	01010101	01011010	01101001	10101010	00001111	35 29 39 25 48 18 44 22	260|260|260/260|
01010101	10101010	10100101	10010110	10101010	11110000	32 34 28 38 19 45 23 41	260|260|260/260|
*32	*16	*8	*4	*2	*1		

```
 1= 0*32 + 0*16 + 0*8 + 0*4 + 0*2 + 0*1 + 1;
63= 1*32 + 1*16 + 1*8 + 1*4 + 1*2 + 0*1 + 1;
51= 1*32 + 1*16 + 0*8 + 0*4 + 1*2 + 0*1 + 1;
```

. . . . .
Vn= An*32 + Bn*16 + Cn*8 + Dn*4 + En*2 + Fn*1 + 1;
(**n** means corresponding position of each unit. V means the value in Classical Notation)

We always use this expression when we compose an object solution by the 6 units.
This equation is mathematically equivalent to the next one below demonstrating the serial multiplications corresponding to the Serial Divisions of Binary Decompositions.
Vn=((((An*2 + Bn)*2 + Cn)*2 + Dn)*2 + En)*2 + Fn + 1;

Definition (1): In each unit every row, column and pan-diagonal must be made of: {0,0,0,0,1,1,1,1}, {0,0,0,1,0,1,1,1}, {0,0,0,1,1,0,1,1}, {0,0,0,1,1,1,0,1}, {0,0,0,1,1,1,1,0}, {0,0,1,0,0,1,1,1}, {0,0,1,0,1,0,1,1}, ... , {1,1,1,1,0,0,0,0} using '0' four times and '1' as often as '0'.
Therefore each sum of those rows, columns and pan-diagonals is calculated in the same style as:
{0+0+0+0+1+1+1+1}*32+{0+0+0+0+1+1+1+1}*16+{0+0+0+0+1+1+1+1}*8
+{0+0+0+0+1+1+1+1}*4+{0+0+0+0+1+1+1+1}*2+{0+0+0+0+1+1+1+1}*1
= {0+0+0+0+1+1+1+1}*(32+16+8+4+2+1) = 4*63 = 252(Decimal)
This makes every sum take the same value. It means that 'Magic Constant' is consequently realized.
252(Decimal) here is calculated and written for the 'Mathematical Notation', which is made of the series of Integers 0~63. It is equivalent to 260(Decimal) of Classical Notation, which is made of the series of Natural Numbers 1~64.

Def (2): Each Unit must consist of 32 digits of '0' and as many '1' as '0'.
This property is always true, whenever the **first** **Definition (1)** is true.

Def (3): Every combination of 6 values of the corresponding positions must be any one of {00000000, 00000001, 00000010, 00000011, 00000101, ..., 00001101, 00001110, 00001111, ..., 11111110, 11111111(N2i)}, and neither repeating usage nor drop-off of any value must be taken.
This condition is logically equivalent to one of the most basic promises for our Classical Notation that we must use the series of Natural Numbers 1~64 to make the objective magic square and use each number strictly once and must not use anyone twice or more often in the same solution.

They say Legendary Leonhard Euler(1707-1783) did not mention even about the pan-diagonals of 'Greco-Latin Squares' (Dr. Mutsumi Suzuki once taught me about this), but we want all pan-diagonals to accept the **Definition (1)** of 'C.E.S.' above and I want to call the "**Complete Euler Squares**" of Order 8 for those which take them all as true.

2. Our Purpose

What we want to do now is to compose various types of pan-diagonal magic squares of order 8 only by these definitions above and directly by the Binary Number System.

We need to have the complete set of all appropriate Units and pick up the 6 actual ones out of them to combine and make each solution by our calcultaions.
We need to know how to build those units, and how to pick them up to combine.
(1) First of all, let's build the complete set of necessary Units by ourselves.
(2) Let's choose 6 units to combine and make them act as the set of 6 Binary Units of Decompositions for each solution.
(3) We must know about the actual way how to combine them appropriately to make only the correct solutions.

3. How to Make the set of Binary Units for our Object

I once tried to make the 'universal' units for every type of order 8, but failed in vain.

In the case of Order 4, there is only one type of solution set for all kinds of pan-diagonal magic squares, and there is really only one set of Binary Units to be made for 'Complete Euler Squares' of Order 4.

But for the Order 8, there are several types of pan-diagonal magic squares and their solution sets are really different from one another.

Therefore, we have to design and make each individual set of Binary Units according to its own definition of each type, similar to the one of objective magic square.

From now on it will be more important for us to make the 'case studies' and know very well about how to design and make its own set of Binary Units for each type.

4. How to Reconstruct the 'Composite and Complete' Magic Squares of Order 8

For the first case, let's study about the reconstruction of the 'C&C' MS88.

We already know about this type so well that we can expect we may surely check any step of composing them carefully and know how to do with it appropriately.

4-1. How to design and compose the Binary Units

Let's compose the Binary Units just in the same way as we would often make such an ordinary magic square as usual, but only with '0' and '1' by the Binary System.

The similar structures between the units and the object are the basic presupposition and the fundamental request of our '**Euler Squares**' and the **New Euler's Method**.

Let's take all the conditions below to our units, and call them the "**Euler Units**" from now on.

[Basic Positions]

```
61 62 63 64 57 58 59 60 61 62 63 64 57 58 59 60
           .-----------------------.
n5 n6 n7 n8|n1|n2|n3|n4|n5|n6|n7|n8|n1 n2 n3 n4
           |--+--+--+--+--+--+--+--|
13 14 15 16|n9|10|11|12|13|14|15|16|n9 10 11 12
           |--+--+--+--+--+--+--+--|
21 22 23 24|17|18|19|20|21|22|23|24|17 18 19 20
           |--+--+--+--+--+--+--+--|
29 30 31 32|25|26|27|28|29|30|31|32|25 26 27 28
           |--+--+--+--+--+--+--+--|
37 38 39 40|33|34|35|36|37|38|39|40|33 34 35 36
           |--+--+--+--+--+--+--+--|
45 46 47 48|41|42|43|44|45|46|47|48|41 42 43 44
           |--+--+--+--+--+--+--+--|
53 54 55 56|49|50|51|52|53|54|55|56|49 50 51 52
           |--+--+--+--+--+--+--+--|
61 62 63 64|57|58|59|60|61|62|63|64|57 58 59 60
           '-----------------------'
n5 n6 n7 n8 n1 n2 n3 n4 n5 n6 n7 n8 n1 n2 n3 n4
```

**** 'Composite Conditions': ****

```
n1+n2+n9+n10=2;
n2+n3+n10+n11=2;
n3+n4+n11+n12=2;
n4+n5+n12+n13=2;
n5+n6+n13+n14=2;
n6+n7+n14+n15=2;
n7+n8+n15+n16=2;
n8+n1+n16+n9=2;
n9+n10+n17+n18=2;
n10+n11+n18+n19=2;
n11+n12+n19+n20=2;
n12+n13+n20+n21=2;
n13+n14+n21+n22=2;
n14+n15+n22+n23=2;
n15+n16+n23+n24=2;
n16+n9+n24+n17=2;
n17+n18+n25+n26=2;
n18+n19+n26+n27=2;
n19+n20+n27+n28=2;
n20+n21+n28+n29=2;
```

```
n21+n22+n29+n30=2;   n22+n23+n30+n31=2;   n23+n24+n31+n32=2;
n24+n17+n32+n25=2;   n25+n26+n33+n34=2;   n26+n27+n34+n35=2;
n27+n28+n35+n36=2;   n28+n29+n36+n37=2;   n29+n30+n37+n38=2;
n30+n31+n38+n39=2;   n31+n32+n39+n40=2;   n32+n25+n40+n33=2;   . . . . . .
```

**** Conditions for every Row, Column and Pan-diagonal ****

```
n1+n2+n3+n4+n5+n6+n7+n8=4;          n1+n9+n17+n25+n33+n41+n49+n57=4;
n9+n10+n11+n12+n13+n14+n15+n16=4;   n2+n10+n18+n26+n34+n42+n50+n58=4;
n17+n18+n19+n20+n21+n22+n23+n24=4;  n3+n11+n19+n27+n35+n43+n51+n59=4;
n25+n26+n27+n28+n29+n30+n31+n32=4;  n4+n12+n20+n28+n36+n44+n52+n60=4;
n33+n34+n35+n36+n37+n38+n39+n40=4;  n5+n13+n21+n29+n37+n45+n53+n61=4;
n41+n42+n43+n44+n45+n46+n47+n48=4;  n6+n14+n22+n30+n38+n46+n54+n62=4;
```

```
  n49+n50+n51+n52+n53+n54+n55+n56=4;  n7+n15+n23+n31+n39+n47+n55+n63=4;
  n57+n58+n59+n60+n61+n62+n63+n64=4;  n8+n16+n24+n32+n40+n48+n56+n64=4;

  ** Pan-diagonal Conditions: **
  n1+n10+n19+n28+n37+n46+n55+n64=4;   n1+n16+n23+n30+n37+n44+n51+n58=4;
  n2+n11+n20+n29+n38+n47+n56+n57=4;   n2+n9+n24+n31+n38+n45+n52+n59=4;
  n3+n12+n21+n30+n39+n48+n49+n58=4;   n3+n10+n17+n32+n39+n46+n53+n60=4;
  n4+n13+n22+n31+n40+n41+n50+n59=4;   n4+n11+n18+n25+n40+n47+n54+n61=4;
  n5+n14+n23+n32+n33+n42+n51+n60=4;   n5+n12+n19+n26+n33+n48+n55+n62=4;
  n6+n15+n24+n25+n34+n43+n52+n61=4;   n6+n13+n20+n27+n34+n41+n56+n63=4;
  n7+n16+n17+n26+n35+n44+n53+n62=4;   n7+n14+n21+n28+n35+n42+n49+n64=4;
  n8+n9+n18+n27+n36+n45+n54+n63=4;    n8+n15+n22+n29+n36+n43+n50+n57=4;

  ** Complete Condtions: **
    ** Complementary pairs of 1 must be located as follows **
  n1+n37=1;    n2+n38=1;    n3+n39=1;    n4+n40=1;    n5+n33=1;
  n6+n34=1;    n7+n35=1;    n8+n36=1;    n9+n45=1;    n10+n46=1;
  n11+n47=1;   n12+n48=1;   n13+n41=1;   n14+n42=1;   n15+n43=1;
  n16+n44=1;   n17+n53=1;   n18+n54=1;   n19+n55=1;   n20+n56=1;
  n21+n49=1;   n22+n50=1;   n23+n51=1;   n24+n52=1;   n25+n61=1;
  n26+n62=1;   n27+n63=1;   n28+n64=1;   n29+n57=1;   n30+n58=1;
  n31+n59=1;   n32+n60=1;
```

I took these basic conditions above and wrote my computer program as follows.

I prepared 32 memory arrays of flag sets to watch every row, every column and every pan-diagonal add up to 4, checking how often any value of the position is used. If the value is used less than 4 times, it means 'usable.' But if not, it means 'not usable.' The next step we have to go varies according to the state of flags.

(n1, n37), (n2, n38), (n3, n39), ..., (n32, n60) are the characteristic Complementary Pairs of 'Complete' type of magic squares of order 8. We can define them two by two in the same procedure for each pair as: `n37=1-n1; n38=1-n2; n39=1-n3; ...; n60=1-n32;`

When we set n1 and n37, we have to make the concerned flags count up by one, and when we return back to the same step after doing any job, we have to make them count down by one.

```
//** 'Composite & Complete' Pan-magic Squares of Oder 8: Composing the 'Complete **
//** Euler Squares' of Order 8 by 'New Euler's Method' with Binary Number System **
//** 'CES88CC.c': Dictated by Kanji Setsuda in 2003, 2006, 2009; Revised on **
//**  Mar. 1, 2011 and May 14, 2012; with MacOSX 10.7.4 and Xcode 4.3.2 **
//
#include <stdio.h>
//
long int cnt;
short lcnt, cnt1, cnt3;
short LSM, CC;
short nm[65], uflg[65];
short tlu[33][65];
short mtc[33][33];
short tn[23041][71];
short anm[5][72];
short n1c[65];
//
short bs[5][2], cs[65][2];
//
short u1,u2,u3,u4,u5,u6;
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void stp09(void), stp10(void), stp11(void), stp12(void);
```

```
void stp13(void), stp14(void), stp15(void), stp16(void);
void stp17(void), stp18(void), stp19(void), stp20(void);
void stp21(void), stp22(void), stp23(void), stp24(void);
void stp25(void), stp26(void), stp27(void), stp28(void);
void stp29(void), stp30(void), stp31(void), stp32(void);
void lurecord(void);
//
void prlunit(void);
void cmbcmp(void);
void cmcm2(void), cmcm3(void), cmcm4(void), cmcm5(void), cmcm6(void), cmcm7(void);
void solrecord(void);
void sol3pr(short x, short y, short z);
void pr3sol(short x);
void prn1c(void);
//
//* Main Program *
int main(){
 short m,n;
  printf("\n");
  printf("** 'Composite & Complete' Pan-magic Squares of Order 8:  **\n");
  printf("**  Composing 'Complete Euler Squares' of Order 8 by the **\n");
  printf("**  Binary Number System and the 'New Euler's Method' **\n");
  for(n=0;n<65;n++){nm[n]=0; uflg[n]=0;}
  for(m=0;m<5;m++){for(n=0;n<2;n++){bs[m][n]=0;}}
  for(m=0;m<65;m++){for(n=0;n<2;n++){cs[m][n]=0;}}
  CC=1; lcnt=0;
  stp01();              //Make the Latin Units
  printf("\n");
  printf(" [Latin Units of Binary Decompositions]\n");
  prlunit();            //Print the Latin Units
  printf("\n");
  printf(" ['Composite & Complete' Pan-magic Squares of Order 8 Reconstructed:\n");
  printf("    List of Standard Solutions(n1==1): Used_Units////// Sol_No#|Sum_Checks|]\n");
  LSM=260; cnt=0; cnt1=0; cnt3=0;
  for(n=0;n<65;n++){n1c[n]=0;}
  cmbcmp();             //Combine & Compose
  sol3pr(1,9,1);
  sol3pr(10,99,10);
  sol3pr(100,999,100);
  sol3pr(1000,cnt1,1000);
  if(cnt3>0){pr3sol(cnt3);}
  printf("  . . . . .\n");
  printf(" [Solution Counts: n1==1/Total = %d/%ld]\n",cnt1,cnt);
  printf("\n");
  printf(" [Classify the Counts according to the Value of n1]\n");
  prn1c();
  printf("  . . . . .\n");
  printf(" [OK!]\n");
  printf("** Calculated and Listed by Kanji Setsuda\n");
  printf("    on May 14, 2012 with MacOSX 10.7.4 & Xcode 4.3.2 **\n");
  printf("\n");
 return 0;
}
//
//** Program Modules for Latin Units of 'Complete Euler Squares' **
//** for the 'Composite & Complete' Pan-magic Squares of Order 8 **
//**  by the Binary Number System and the 'New Euler's Method'  **
//**  Listed by Kanji Setsuda on May 14, 2012 with Mac Xcode 4  **
// Level #1:
// Set n1 & n37
void stp01(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((bs[1][a]<4)&&(bs[2][a]<4)&&(bs[3][a]<4)){
      if(bs[3][b]<4){
```

```
        if((cs[1][a]<2)&&(cs[8][a]<2)&&(cs[57][a]<2)&&(cs[64][a]<2)){
          if((cs[28][b]<2)&&(cs[29][b]<2)&&(cs[36][b]<2)&&(cs[37][b]<2)){
            nm[1]=a; nm[37]=b;
            bs[1][a]++; bs[2][a]++; bs[3][a]++; bs[3][b]++;
            cs[1][a]++; cs[8][a]++; cs[57][a]++; cs[64][a]++;
            cs[28][b]++; cs[29][b]++; cs[36][b]++; cs[37][b]++;
            stp02();
            cs[37][b]--; cs[36][b]--; cs[29][b]--; cs[28][b]--;
            cs[64][a]--; cs[57][a]--; cs[8][a]--; cs[1][a]--;
            bs[3][b]--; bs[3][a]--; bs[2][a]--; bs[1][a]--;
          }}}}
  }
}
// Set n2 & n38
void stp02(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if(bs[1][a]<4){
      if((cs[1][a]<2)&&(cs[2][a]<2)&&(cs[57][a]<2)&&(cs[58][a]<2)){
        if((cs[29][b]<2)&&(cs[30][b]<2)&&(cs[37][b]<2)&&(cs[38][b]<2)){
          nm[2]=a; nm[38]=b;
          bs[1][a]++;
          cs[1][a]++; cs[2][a]++; cs[57][a]++; cs[58][a]++;
          cs[29][b]++; cs[30][b]++; cs[37][b]++; cs[38][b]++;
          stp03();
          cs[38][b]--; cs[37][b]--; cs[30][b]--; cs[29][b]--;
          cs[58][a]--; cs[57][a]--; cs[2][a]--; cs[1][a]--;
          bs[1][a]--;
        }}}
  }
}
// Set n9 & n45
void stp03(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if(bs[2][a]<4){
      if((cs[1][a]<2)&&(cs[8][a]<2)&&(cs[9][a]<2)&&(cs[16][a]<2)){
        if((cs[36][b]<2)&&(cs[37][b]<2)&&(cs[44][b]<2)&&(cs[45][b]<2)){
          nm[9]=a; nm[45]=b;
          bs[2][a]++;
          cs[1][a]++; cs[8][a]++; cs[9][a]++; cs[16][a]++;
          cs[36][b]++; cs[37][b]++; cs[44][b]++; cs[45][b]++;
          stp04();
          cs[45][b]--; cs[44][b]--; cs[37][b]--; cs[36][b]--;
          cs[16][a]--; cs[9][a]--; cs[8][a]--; cs[1][a]--;
          bs[2][a]--;
        }}}
  }
}
// Set n10 & n46
void stp04(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[3][a]<4){
      if(bs[3][b]<4){
        if((cs[1][a]<2)&&(cs[2][a]<2)&&(cs[9][a]<2)&&(cs[10][a]<2)){
          if((cs[37][b]<2)&&(cs[38][b]<2)&&(cs[45][b]<2)&&(cs[46][b]<2)){
            nm[10]=a; nm[46]=b;
            bs[3][a]++; bs[3][b]++;
            cs[1][a]++; cs[2][a]++; cs[9][a]++; cs[10][a]++;
            cs[37][b]++; cs[38][b]++; cs[45][b]++; cs[46][b]++;
            stp05();
            cs[46][b]--; cs[45][b]--; cs[38][b]--; cs[37][b]--;
            cs[10][a]--; cs[9][a]--; cs[2][a]--; cs[1][a]--;
            bs[3][b]--; bs[3][a]--;
```

```
          }}}}
  }
}
// Set n3 & n39
void stp05(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[1][a]<4){
      if((cs[2][a]<2)&&(cs[3][a]<2)&&(cs[58][a]<2)&&(cs[59][a]<2)){
        if((cs[30][b]<2)&&(cs[31][b]<2)&&(cs[38][b]<2)&&(cs[39][b]<2)){
          nm[3]=a; nm[39]=b;
          bs[1][a]++;
          cs[2][a]++; cs[3][a]++; cs[58][a]++; cs[59][a]++;
          cs[30][b]++; cs[31][b]++; cs[38][b]++; cs[39][b]++;
          stp06();
          cs[39][b]--; cs[38][b]--; cs[31][b]--; cs[30][b]--;
          cs[59][a]--; cs[58][a]--; cs[3][a]--; cs[2][a]--;
          bs[1][a]--;
        }}}
  }
}
// Set n11 & n47
void stp06(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[2][a]<2)&&(cs[3][a]<2)&&(cs[10][a]<2)&&(cs[11][a]<2)){
      if((cs[38][b]<2)&&(cs[39][b]<2)&&(cs[46][b]<2)&&(cs[47][b]<2)){
        nm[11]=a; nm[47]=b;
        cs[2][a]++; cs[3][a]++; cs[10][a]++; cs[11][a]++;
        cs[38][b]++; cs[39][b]++; cs[46][b]++; cs[47][b]++;
        stp07();
        cs[47][b]--; cs[46][b]--; cs[39][b]--; cs[38][b]--;
        cs[11][a]--; cs[10][a]--; cs[3][a]--; cs[2][a]--;
      }}
  }
}
// Set n4 & n40
void stp07(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if(bs[1][a]<4){
      if((cs[3][a]<2)&&(cs[4][a]<2)&&(cs[59][a]<2)&&(cs[60][a]<2)){
        if((cs[31][b]<2)&&(cs[32][b]<2)&&(cs[39][b]<2)&&(cs[40][b]<2)){
          nm[4]=a; nm[40]=b;
          bs[1][a]++;
          cs[3][a]++; cs[4][a]++; cs[59][a]++; cs[60][a]++;
          cs[31][b]++; cs[32][b]++; cs[39][b]++; cs[40][b]++;
          stp08();
          cs[40][b]--; cs[39][b]--; cs[32][b]--; cs[31][b]--;
          cs[60][a]--; cs[59][a]--; cs[4][a]--; cs[3][a]--;
          bs[1][a]--;
        }}}
  }
}
// Set n12 & n48
void stp08(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[3][a]<2)&&(cs[4][a]<2)&&(cs[11][a]<2)&&(cs[12][a]<2)){
      if((cs[39][b]<2)&&(cs[40][b]<2)&&(cs[47][b]<2)&&(cs[48][b]<2)){
        nm[12]=a; nm[48]=b;
        cs[3][a]++; cs[4][a]++; cs[11][a]++; cs[12][a]++;
        cs[39][b]++; cs[40][b]++; cs[47][b]++; cs[48][b]++;
        stp09();
        cs[48][b]--; cs[47][b]--; cs[40][b]--; cs[39][b]--;
```

```
          cs[12][a]--; cs[11][a]--; cs[4][a]--; cs[3][a]--;
        }}
    }
}
// Set n5 & n33
void stp09(){
 short a,b;
   for(a=0;a<2;a++){b=CC-a;
     if(bs[1][a]<4){
       if(bs[2][b]<4){
         if((cs[4][a]<2)&&(cs[5][a]<2)&&(cs[60][a]<2)&&(cs[61][a]<2)){
           if((cs[25][b]<2)&&(cs[32][b]<2)&&(cs[33][b]<2)&&(cs[40][b]<2)){
             nm[5]=a; nm[33]=b;
             bs[1][a]++; bs[2][b]++;
             cs[4][a]++; cs[5][a]++; cs[60][a]++; cs[61][a]++;
             cs[25][b]++; cs[32][b]++; cs[33][b]++; cs[40][b]++;
             stp10();
             cs[40][b]--; cs[33][b]--; cs[32][b]--; cs[25][b]--;
             cs[61][a]--; cs[60][a]--; cs[5][a]--; cs[4][a]--;
             bs[2][b]--; bs[1][a]--;
           }}}}
   }
}
// Set n13 & n41
void stp10(){
 short a,b;
   for(a=0;a<2;a++){b=CC-a;
     if(bs[2][b]<4){
       if((cs[4][a]<2)&&(cs[5][a]<2)&&(cs[12][a]<2)&&(cs[13][a]<2)){
         if((cs[33][b]<2)&&(cs[40][b]<2)&&(cs[41][b]<2)&&(cs[48][b]<2)){
           nm[13]=a; nm[41]=b;
           bs[2][b]++;
           cs[4][a]++; cs[5][a]++; cs[12][a]++; cs[13][a]++;
           cs[33][b]++; cs[40][b]++; cs[41][b]++; cs[48][b]++;
           stp11();
           cs[48][b]--; cs[41][b]--; cs[40][b]--; cs[33][b]--;
           cs[13][a]--; cs[12][a]--; cs[5][a]--; cs[4][a]--;
           bs[2][b]--;
         }}}
   }
}
// Set n6 & n34
void stp11(){
 short a,b;
   for(a=1;a>=0;a--){b=CC-a;
     if(bs[1][a]<4){
       if((cs[5][a]<2)&&(cs[6][a]<2)&&(cs[61][a]<2)&&(cs[62][a]<2)){
         if((cs[25][b]<2)&&(cs[26][b]<2)&&(cs[33][b]<2)&&(cs[34][b]<2)){
           nm[6]=a; nm[34]=b;
           bs[1][a]++;
           cs[5][a]++; cs[6][a]++; cs[61][a]++; cs[62][a]++;
           cs[25][b]++; cs[26][b]++; cs[33][b]++; cs[34][b]++;
           stp12();
           cs[34][b]--; cs[33][b]--; cs[26][b]--; cs[25][b]--;
           cs[62][a]--; cs[61][a]--; cs[6][a]--; cs[5][a]--;
           bs[1][a]--;
         }}}
   }
}
// Set n14 & n42
void stp12(){
 short a,b;
   for(a=0;a<2;a++){b=CC-a;
     if((cs[5][a]<2)&&(cs[6][a]<2)&&(cs[13][a]<2)&&(cs[14][a]<2)){
       if((cs[33][b]<2)&&(cs[34][b]<2)&&(cs[41][b]<2)&&(cs[42][b]<2)){
```

```
          nm[14]=a; nm[42]=b;
          cs[5][a]++; cs[6][a]++; cs[13][a]++; cs[14][a]++;
          cs[33][b]++; cs[34][b]++; cs[41][b]++; cs[42][b]++;
          stp13();
          cs[42][b]--; cs[41][b]--; cs[34][b]--; cs[33][b]--;
          cs[14][a]--; cs[13][a]--; cs[6][a]--; cs[5][a]--;
        }}
  }
}
// Set n7 & n35
void stp13(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[1][a]<4){
      if((cs[6][a]<2)&&(cs[7][a]<2)&&(cs[62][a]<2)&&(cs[63][a]<2)){
        if((cs[26][b]<2)&&(cs[27][b]<2)&&(cs[34][b]<2)&&(cs[35][b]<2)){
          nm[7]=a; nm[35]=b;
          bs[1][a]++;
          cs[6][a]++; cs[7][a]++; cs[62][a]++; cs[63][a]++;
          cs[26][b]++; cs[27][b]++; cs[34][b]++; cs[35][b]++;
          stp14();
          cs[35][b]--; cs[34][b]--; cs[27][b]--; cs[26][b]--;
          cs[63][a]--; cs[62][a]--; cs[7][a]--; cs[6][a]--;
          bs[1][a]--;
        }}}
  }
}
// Set n15 & n43
void stp14(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[4][a]<4){
      if(bs[4][b]<4){
        if((cs[6][a]<2)&&(cs[7][a]<2)&&(cs[14][a]<2)&&(cs[15][a]<2)){
          if((cs[34][b]<2)&&(cs[35][b]<2)&&(cs[42][b]<2)&&(cs[43][b]<2)){
            nm[15]=a; nm[43]=b;
            bs[4][a]++; bs[4][b]++;
            cs[6][a]++; cs[7][a]++; cs[14][a]++; cs[15][a]++;
            cs[34][b]++; cs[35][b]++; cs[42][b]++; cs[43][b]++;
            stp15();
            cs[43][b]--; cs[42][b]--; cs[35][b]--; cs[34][b]--;
            cs[15][a]--; cs[14][a]--; cs[7][a]--; cs[6][a]--;
            bs[4][b]--; bs[4][a]--;
          }}}}
  }
}
// Set n8 & n36
void stp15(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if((bs[1][a]<4)&&(bs[4][a]<4)){
      if(bs[4][b]<4){
        if((cs[7][a]<2)&&(cs[8][a]<2)&&(cs[63][a]<2)&&(cs[64][a]<2)){
          if((cs[27][b]<2)&&(cs[28][b]<2)&&(cs[35][b]<2)&&(cs[36][b]<2)){
            nm[8]=a; nm[36]=b;
            bs[1][a]++; bs[4][a]++; bs[4][b]++;
            cs[7][a]++; cs[8][a]++; cs[63][a]++; cs[64][a]++;
            cs[27][b]++; cs[28][b]++; cs[35][b]++; cs[36][b]++;
            stp16();
            cs[36][b]--; cs[35][b]--; cs[28][b]--; cs[27][b]--;
            cs[64][a]--; cs[63][a]--; cs[8][a]--; cs[7][a]--;
            bs[4][b]--; bs[4][a]--; bs[1][a]--;
          }}}}
  }
}
```

```
// Set n16 & n44
void stp16(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[7][a]<2)&&(cs[8][a]<2)&&(cs[15][a]<2)&&(cs[16][a]<2)){
      if((cs[35][b]<2)&&(cs[36][b]<2)&&(cs[43][b]<2)&&(cs[44][b]<2)){
        nm[16]=a; nm[44]=b;
        cs[7][a]++; cs[8][a]++; cs[15][a]++; cs[16][a]++;
        cs[35][b]++; cs[36][b]++; cs[43][b]++; cs[44][b]++;
        stp17();
        cs[44][b]--; cs[43][b]--; cs[36][b]--; cs[35][b]--;
        cs[16][a]--; cs[15][a]--; cs[8][a]--; cs[7][a]--;
      }}
  }
}
// Level #2:
// Set n17 & n53
void stp17(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[2][a]<4){
      if((cs[9][a]<2)&&(cs[16][a]<2)&&(cs[17][a]<2)&&(cs[24][a]<2)){
        if((cs[44][b]<2)&&(cs[45][b]<2)&&(cs[52][b]<2)&&(cs[53][b]<2)){
          nm[17]=a; nm[53]=b;
          bs[2][a]++;
          cs[9][a]++; cs[16][a]++; cs[17][a]++; cs[24][a]++;
          cs[44][b]++; cs[45][b]++; cs[52][b]++; cs[53][b]++;
          stp18();
          cs[53][b]--; cs[52][b]--; cs[45][b]--; cs[44][b]--;
          cs[24][a]--; cs[17][a]--; cs[16][a]--; cs[9][a]--;
          bs[2][a]--;
        }}}
  }
}
// Set n18 & n54
void stp18(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[9][a]<2)&&(cs[10][a]<2)&&(cs[17][a]<2)&&(cs[18][a]<2)){
      if((cs[45][b]<2)&&(cs[46][b]<2)&&(cs[53][b]<2)&&(cs[54][b]<2)){
        nm[18]=a; nm[54]=b;
        cs[9][a]++; cs[10][a]++; cs[17][a]++; cs[18][a]++;
        cs[45][b]++; cs[46][b]++; cs[53][b]++; cs[54][b]++;
        stp19();
        cs[54][b]--; cs[53][b]--; cs[46][b]--; cs[45][b]--;
        cs[18][a]--; cs[17][a]--; cs[10][a]--; cs[9][a]--;
      }}
  }
}
// Set n19 & n55
void stp19(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[3][a]<4){
      if(bs[3][b]<4){
        if((cs[10][a]<2)&&(cs[11][a]<2)&&(cs[18][a]<2)&&(cs[19][a]<2)){
          if((cs[46][b]<2)&&(cs[47][b]<2)&&(cs[54][b]<2)&&(cs[55][b]<2)){
            nm[19]=a; nm[55]=b;
            bs[3][a]++; bs[3][b]++;
            cs[10][a]++; cs[11][a]++; cs[18][a]++; cs[19][a]++;
            cs[46][b]++; cs[47][b]++; cs[54][b]++; cs[55][b]++;
            stp20();
            cs[55][b]--; cs[54][b]--; cs[47][b]--; cs[46][b]--;
            cs[19][a]--; cs[18][a]--; cs[11][a]--; cs[10][a]--;
            bs[3][b]--; bs[3][a]--;
```

```
            }}}}
  }
}
// Set n20 & n56
void stp20(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[11][a]<2)&&(cs[12][a]<2)&&(cs[19][a]<2)&&(cs[20][a]<2)){
      if((cs[47][b]<2)&&(cs[48][b]<2)&&(cs[55][b]<2)&&(cs[56][b]<2)){
        nm[20]=a; nm[56]=b;
        cs[11][a]++; cs[12][a]++; cs[19][a]++; cs[20][a]++;
        cs[47][b]++; cs[48][b]++; cs[55][b]++; cs[56][b]++;
        stp21();
        cs[56][b]--; cs[55][b]--; cs[48][b]--; cs[47][b]--;
        cs[20][a]--; cs[19][a]--; cs[12][a]--; cs[11][a]--;
      }}
  }
}
// Set n21 & n49
void stp21(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[2][b]<4){
      if((cs[12][a]<2)&&(cs[13][a]<2)&&(cs[20][a]<2)&&(cs[21][a]<2)){
        if((cs[41][b]<2)&&(cs[48][b]<2)&&(cs[49][b]<2)&&(cs[56][b]<2)){
          nm[21]=a; nm[49]=b;
          bs[2][b]++;
          cs[12][a]++; cs[13][a]++; cs[20][a]++; cs[21][a]++;
          cs[41][b]++; cs[48][b]++; cs[49][b]++; cs[56][b]++;
          stp22();
          cs[56][b]--; cs[49][b]--; cs[48][b]--; cs[41][b]--;
          cs[21][a]--; cs[20][a]--; cs[13][a]--; cs[12][a]--;
          bs[2][b]--;
        }}}
  }
}
// Set n22 & n50
void stp22(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[4][a]<4){
      if(bs[4][b]<4){
        if((cs[13][a]<2)&&(cs[14][a]<2)&&(cs[21][a]<2)&&(cs[22][a]<2)){
          if((cs[41][b]<2)&&(cs[42][b]<2)&&(cs[49][b]<2)&&(cs[50][b]<2)){
            nm[22]=a; nm[50]=b;
            bs[4][a]++; bs[4][b]++;
            cs[13][a]++; cs[14][a]++; cs[21][a]++; cs[22][a]++;
            cs[41][b]++; cs[42][b]++; cs[49][b]++; cs[50][b]++;
            stp23();
            cs[50][b]--; cs[49][b]--; cs[42][b]--; cs[41][b]--;
            cs[22][a]--; cs[21][a]--; cs[14][a]--; cs[13][a]--;
            bs[4][b]--; bs[4][a]--;
          }}}}
  }
}
// Set n23 & n51
void stp23(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[14][a]<2)&&(cs[15][a]<2)&&(cs[22][a]<2)&&(cs[23][a]<2)){
      if((cs[42][b]<2)&&(cs[43][b]<2)&&(cs[50][b]<2)&&(cs[51][b]<2)){
        nm[23]=a; nm[51]=b;
        cs[14][a]++; cs[15][a]++; cs[22][a]++; cs[23][a]++;
        cs[42][b]++; cs[43][b]++; cs[50][b]++; cs[51][b]++;
        stp24();
```

```
          cs[51][b]--; cs[50][b]--; cs[43][b]--; cs[42][b]--;
          cs[23][a]--; cs[22][a]--; cs[15][a]--; cs[14][a]--;
        }}
  }
}
// Set n24 & n52
void stp24(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[15][a]<2)&&(cs[16][a]<2)&&(cs[23][a]<2)&&(cs[24][a]<2)){
      if((cs[43][b]<2)&&(cs[44][b]<2)&&(cs[51][b]<2)&&(cs[52][b]<2)){
        nm[24]=a; nm[52]=b;
        cs[15][a]++; cs[16][a]++; cs[23][a]++; cs[24][a]++;
        cs[43][b]++; cs[44][b]++; cs[51][b]++; cs[52][b]++;
        stp25();
        cs[52][b]--; cs[51][b]--; cs[44][b]--; cs[43][b]--;
        cs[24][a]--; cs[23][a]--; cs[16][a]--; cs[15][a]--;
      }}
  }
}
// Level #3:
// Set n25 & n61
void stp25(){
 short a,b;
  for(a=1;a>=0;a--){b=CC-a;
    if(bs[2][a]<4){
      if((cs[17][a]<2)&&(cs[24][a]<2)&&(cs[25][a]<2)&&(cs[32][a]<2)){
        if((cs[52][b]<2)&&(cs[53][b]<2)&&(cs[60][b]<2)&&(cs[61][b]<2)){
          nm[25]=a; nm[61]=b;
          bs[2][a]++;
          cs[17][a]++; cs[24][a]++; cs[25][a]++; cs[32][a]++;
          cs[52][b]++; cs[53][b]++; cs[60][b]++; cs[61][b]++;
          stp26();
          cs[61][b]--; cs[60][b]--; cs[53][b]--; cs[52][b]--;
          cs[32][a]--; cs[25][a]--; cs[24][a]--; cs[17][a]--;
          bs[2][a]--;
        }}}
  }
}
// Set n26 & n62
void stp26(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[17][a]<2)&&(cs[18][a]<2)&&(cs[25][a]<2)&&(cs[26][a]<2)){
      if((cs[53][b]<2)&&(cs[54][b]<2)&&(cs[61][b]<2)&&(cs[62][b]<2)){
        nm[26]=a; nm[62]=b;
        cs[17][a]++; cs[18][a]++; cs[25][a]++; cs[26][a]++;
        cs[53][b]++; cs[54][b]++; cs[61][b]++; cs[62][b]++;
        stp27();
        cs[62][b]--; cs[61][b]--; cs[54][b]--; cs[53][b]--;
        cs[26][a]--; cs[25][a]--; cs[18][a]--; cs[17][a]--;
      }}
  }
}
// Set n27 & n63
void stp27(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[18][a]<2)&&(cs[19][a]<2)&&(cs[26][a]<2)&&(cs[27][a]<2)){
      if((cs[54][b]<2)&&(cs[55][b]<2)&&(cs[62][b]<2)&&(cs[63][b]<2)){
        nm[27]=a; nm[63]=b;
        cs[18][a]++; cs[19][a]++; cs[26][a]++; cs[27][a]++;
        cs[54][b]++; cs[55][b]++; cs[62][b]++; cs[63][b]++;
        stp28();
        cs[63][b]--; cs[62][b]--; cs[55][b]--; cs[54][b]--;
```

```
        cs[27][a]--; cs[26][a]--; cs[19][a]--; cs[18][a]--;
      }}
  }
}
// Set n28 & n64
void stp28(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[3][a]<4){
      if(bs[3][b]<4){
        if((cs[19][a]<2)&&(cs[20][a]<2)&&(cs[27][a]<2)&&(cs[28][a]<2)){
          if((cs[55][b]<2)&&(cs[56][b]<2)&&(cs[63][b]<2)&&(cs[64][b]<2)){
            nm[28]=a; nm[64]=b;
            bs[3][a]++; bs[3][b]++;
            cs[19][a]++; cs[20][a]++; cs[27][a]++; cs[28][a]++;
            cs[55][b]++; cs[56][b]++; cs[63][b]++; cs[64][b]++;
            stp29();
            cs[64][b]--; cs[63][b]--; cs[56][b]--; cs[55][b]--;
            cs[28][a]--; cs[27][a]--; cs[20][a]--; cs[19][a]--;
            bs[3][b]--; bs[3][a]--;
          }}}}
  }
}
// Set n29 & n57
void stp29(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if(bs[4][a]<4){
      if((bs[2][b]<4)&&(bs[4][b]<4)){
        if((cs[20][a]<2)&&(cs[21][a]<2)&&(cs[28][a]<2)&&(cs[29][a]<2)){
          if((cs[49][b]<2)&&(cs[56][b]<2)&&(cs[57][b]<2)&&(cs[64][b]<2)){
            nm[29]=a; nm[57]=b;
            bs[4][a]++; bs[2][b]++; bs[4][b]++;
            cs[20][a]++; cs[21][a]++; cs[28][a]++; cs[29][a]++;
            cs[49][b]++; cs[56][b]++; cs[57][b]++; cs[64][b]++;
            stp30();
            cs[64][b]--; cs[57][b]--; cs[56][b]--; cs[49][b]--;
            cs[29][a]--; cs[28][a]--; cs[21][a]--; cs[20][a]--;
            bs[4][b]--; bs[2][b]--; bs[4][a]--;
          }}}}
  }
}
// Set n30 & n58
void stp30(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[21][a]<2)&&(cs[22][a]<2)&&(cs[29][a]<2)&&(cs[30][a]<2)){
      if((cs[49][b]<2)&&(cs[50][b]<2)&&(cs[57][b]<2)&&(cs[58][b]<2)){
        nm[30]=a; nm[58]=b;
        cs[21][a]++; cs[22][a]++; cs[29][a]++; cs[30][a]++;
        cs[49][b]++; cs[50][b]++; cs[57][b]++; cs[58][b]++;
        stp31();
        cs[58][b]--; cs[57][b]--; cs[50][b]--; cs[49][b]--;
        cs[30][a]--; cs[29][a]--; cs[22][a]--; cs[21][a]--;
      }}
  }
}
// Set n31 & n59
void stp31(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[22][a]<2)&&(cs[23][a]<2)&&(cs[30][a]<2)&&(cs[31][a]<2)){
      if((cs[50][b]<2)&&(cs[51][b]<2)&&(cs[58][b]<2)&&(cs[59][b]<2)){
        nm[31]=a; nm[59]=b;
        cs[22][a]++; cs[23][a]++; cs[30][a]++; cs[31][a]++;
```

```
          cs[50][b]++; cs[51][b]++; cs[58][b]++; cs[59][b]++;
          stp32();
          cs[59][b]--; cs[58][b]--; cs[51][b]--; cs[50][b]--;
          cs[31][a]--; cs[30][a]--; cs[23][a]--; cs[22][a]--;
        }}
    }
}
// Set n32 & n60
void stp32(){
 short a,b;
  for(a=0;a<2;a++){b=CC-a;
    if((cs[23][a]<2)&&(cs[24][a]<2)&&(cs[31][a]<2)&&(cs[32][a]<2)){
      if((cs[51][b]<2)&&(cs[52][b]<2)&&(cs[59][b]<2)&&(cs[60][b]<2)){
        nm[32]=a; nm[60]=b;
        cs[23][a]++; cs[24][a]++; cs[31][a]++; cs[32][a]++;
        cs[51][b]++; cs[52][b]++; cs[59][b]++; cs[60][b]++;
        lurecord();
        cs[60][b]--; cs[59][b]--; cs[52][b]--; cs[51][b]--;
        cs[32][a]--; cs[31][a]--; cs[24][a]--; cs[23][a]--;
      }}
    }
}
//
//* Record the Latin Units *
void lurecord(){
 short n;
  tlu[lcnt][0]=lcnt+1;
  for(n=1;n<65;n++){tlu[lcnt][n]=nm[n];}
  lcnt++;
}
//
//* Classify and Print the Latin Units *
void prlunit(){
 short t,m,n,l;
 short l8;
  for(m=0;m<lcnt;m++){
    for(n=0;n<lcnt;n++){
      t=0;
      for(l=1;l<65;l++){if(tlu[m][l]==tlu[n][l]){t++;}}
      mtc[m][n]=t;
    }
  }
  for(t=0;t<lcnt;t=t+8){
    printf("%9d/%9d/%9d/%9d/%9d/%9d/%9d/%9d/\n",
           t+1,t+2,t+3,t+4,t+5,t+6,t+7,t+8);
    for(l=0;l<8;l++){l8=l*8;
      for(m=t;m<(t+8);m++){
        printf("  ");
        for(n=1;n<9;n++){printf("%d",tlu[m][l8+n]);}
      }
      printf("\n");
    }
  }
  printf(" [Count of Latin Units = %d]\n",lcnt);
  printf("\n");
  printf(" [Reference Table for the Best Match: 32 is the Best;]\n");
  printf("  *|");
  for(n=1;n<=lcnt;n++){printf("%3d",n);} printf("\n");
  printf(" --+");
  for(n=0;n<lcnt;n++){printf("---");}; printf("\n");
  for(m=0;m<lcnt;m++){
    printf("%3d|",m+1);
    for(n=0;n<lcnt;n++){t=mtc[m][n];
      if(t>=0){printf("%3d",t);}else{printf("  -");}}
    printf("\n");
```

```
  }
  printf(" --+");
  for(n=0;n<lcnt;n++){printf("---");}; printf("\n");
  printf("  *|");
  for(n=1;n<=lcnt;n++){printf("%3d",n);} printf("\n");
}
//
//* Combine & Compose *
void cmbcmp(){
  for(u1=0;u1<(lcnt/2);u1++){
    cmcm2();}
}
//* Sub #2 for 'Combine & Compose' *
void cmcm2(){
  for(u2=0;u2<lcnt;u2++){
    if(mtc[u2][u1]==32){
      cmcm3();}
  }
}
//* Sub #3 for 'Combine & Compose' *
void cmcm3(){
  for(u3=0;u3<lcnt;u3++){
    if(mtc[u3][u1]==32){
      if(mtc[u3][u2]==32){
        cmcm4();}
    }}
}
//* Sub #4 for 'Combine & Compose' *
void cmcm4(){
  for(u4=0;u4<lcnt;u4++){
    if(mtc[u4][u1]==32){
      if(mtc[u4][u2]==32){
        if(mtc[u4][u3]==32){cmcm5();}
      }}}
}
//* Sub #5 for 'Combine & Compose' *
void cmcm5(){
  for(u5=0;u5<lcnt;u5++){
    if(mtc[u5][u1]==32){
      if(mtc[u5][u2]==32){
        if(mtc[u5][u3]==32){
          if(mtc[u5][u4]==32){cmcm6();}
        }}}}
}
//* Sub #6 for 'Combine & Compose' *
void cmcm6(){
  for(u6=0;u6<lcnt;u6++){
    if(mtc[u6][u1]==32){
      if(mtc[u6][u2]==32){
        if(mtc[u6][u3]==32){
          if(mtc[u6][u4]==32){
            if(mtc[u6][u5]==32){cmcm7();}
          }}}}}
}
//* Sub #7 for 'Combine & Compose' *
void cmcm7(){
 short n,dt,fc;
  for(n=1;n<65;n++){uflg[n]=0;}
  for(n=1;n<65;n++){
    dt=tlu[u1][n]*32+tlu[u2][n]*16+tlu[u3][n]*8+tlu[u4][n]*4 +tlu[u5][n]*2+tlu[u6][n]+1;
    nm[n]=dt; uflg[dt]++;}
  fc=0;
  for(n=1;n<65;n++){if(uflg[n]==1){fc++;}else{break;}}
  if(fc==64){
    if((nm[1]<nm[64])&&(nm[1]<nm[57])&&(nm[57]<nm[8])){solrecord();}
```

```
  }
}
//
//* Record the Answers only with n1=1 *
void solrecord(){
 short n;
  cnt++; n1c[nm[1]]++;
  if(nm[1]==1){
    tn[cnt1][0]=cnt1+1;
    for(n=1;n<65;n++){tn[cnt1][n]=nm[n];}
    tn[cnt1][65]=u1; tn[cnt1][66]=u2; tn[cnt1][67]=u3;
    tn[cnt1][68]=u4; tn[cnt1][69]=u5; tn[cnt1][70]=u6;
    cnt1++;}
}
//
//* Print All Solutions with n1==1 *
void sol3pr(short x, short y, short z){
 short m,n;
  for(m=x-1;m<y;m=m+z){
    anm[cnt3][0]=m+1;
    for(n=1;n<65;n++){anm[cnt3][n]=tn[m][n];}
    for(n=65;n<71;n++){anm[cnt3][n]=tn[m][n]+1;}
    cnt3++;
    if(cnt3==3){pr3sol(cnt3); cnt3=0;}
  }
}
//
//* Print 3 Solutions in 1 line *
void pr3sol(short x){
 short l,l8,m,n;
  for(m=0;m<x;m++){
    printf("%3d/%2d/%2d/%2d/%2d/%2d/%6d#",
            anm[m][65],anm[m][66],anm[m][67],anm[m][68],anm[m][69],anm[m][70],anm[m][0]);
    if(m+1<x){printf(" ");}
  }
  printf("\n");
  for(l=0;l<8;l++){l8=l*8;
    for(m=0;m<x;m++){
      printf(" ");
      for(n=1;n<9;n++){printf("%3d",anm[m][l8+n]);}
      if(m+1<x){printf("  ");}
    }
    printf("\n");
  }
}
//
//* Print the Count according to n1 *
void prn1c(){
 short m;
  for(m=1;m<37;m++){
    printf("  [%2d]%6d,",m,n1c[m]);
    if(m%6==0){printf("\n");}
  }
}
//
```

All answers are stored in the memory array `tlu[33][65]`, so that you can see and use the data for another purpose any time you want to.

The next list shows all the 32 Euler Units I could get by this program.

```
** 'Composite & Complete' Pan-magic Squares of Order 8:  **
**  Composing 'Complete Euler Squares' of Order 8 by the **
**  Binary Number System and the 'New Euler's Method' **
```

[List of **Euler Units** by Binary Decompositions]

1/	2/	3/	4/	5/	6/	7/	8/
01010101	01010101	01010101	01010101	01011010	01001011	01111000	01101001
10101010	10101010	10101010	10101010	10100101	10110100	10000111	10010110
01010101	01010101	10101010	10101010	01011010	01001011	01111000	01101001
10101010	01010101	10101010	01010101	10100101	10110100	10000111	10010110
10101010	10101010	10101010	10101010	01011010	01001011	01111000	01101001
01010101	01010101	01010101	01010101	10100101	10110100	10000111	10010110
10101010	10101010	01010101	01010101	01011010	01001011	01111000	01101001
01010101	10101010	01010101	10101010	10100101	10110100	10000111	10010110

9/	10/	11/	12/	13/	14/	15/	16/
01010101	01010101	01010101	01010101	00011110	00001111	00111100	00101101
01010101	01010101	01010101	01010101	11100001	11110000	11000011	11010010
01010101	01010101	10101010	10101010	00011110	00001111	00111100	00101101
10101010	01010101	10101010	01010101	11100001	11110000	11000011	11010010
10101010	10101010	10101010	10101010	00011110	00001111	00111100	00101101
10101010	10101010	10101010	10101010	11100001	11110000	11000011	11010010
10101010	10101010	01010101	01010101	00011110	00001111	00111100	00101101
01010101	10101010	01010101	10101010	11100001	11110000	11000011	11010010

17/	18/	19/	20/	21/	22/	23/	24/
11010010	11000011	11110000	11100001	10101010	10101010	10101010	10101010
00101101	00111100	00001111	00011110	10101010	10101010	10101010	10101010
11010010	11000011	11110000	11100001	01010101	01010101	10101010	10101010
00101101	00111100	00001111	00011110	10101010	01010101	10101010	01010101
11010010	11000011	11110000	11100001	01010101	01010101	01010101	01010101
00101101	00111100	00001111	00011110	01010101	01010101	01010101	01010101
11010010	11000011	11110000	11100001	10101010	10101010	01010101	01010101
00101101	00111100	00001111	00011110	01010101	10101010	01010101	10101010

25/	26/	27/	28/	29/	30/	31/	32/
10010110	10000111	10110100	10100101	10101010	10101010	10101010	10101010
01101001	01111000	01001011	01011010	01010101	01010101	01010101	01010101
10010110	10000111	10110100	10100101	01010101	01010101	10101010	10101010
01101001	01111000	01001011	01011010	10101010	01010101	10101010	01010101
10010110	10000111	10110100	10100101	01010101	01010101	01010101	01010101
01101001	01111000	01001011	01011010	10101010	10101010	10101010	10101010
10010110	10000111	10110100	10100101	10101010	10101010	01010101	01010101
01101001	01111000	01001011	01011010	01010101	10101010	01010101	10101010

[Count of Euler Units = 32]

4-2. How to Pick up and Combine the 6 Euler Units to Compose our Object

We have got so many Latin Squares as 32. How can we choose 6 units to combine and compose each 'Composite & Complete' Magic Squares of Order 8?

$32^6 = 32 * 32 * 32 * 32 * 32 * 32 = 1073741824$

Do we have to examine so many answers as one billion, know whether each of them is correct or not? It will surely take too long a time for us to go on.

We expect correct solutions be counted no more than 368640. I am afraid too many wrong answers should be produced in the process of combining without any selection or examination in advance.

[Samples of **Wrong Solutions**: Used Units ////// Sol_No.]

1/ 1/ 1/ 4/ 6/ 7/ ??	1/ 2/ 1/ 9/12/18/ ??	1/ 3/ 1/ 9/12/18/ ??
1 64 1 64 1 64 1 64	2 64 4 62 7 57 5 59	2 64 4 62 7 57 5 59
61 4 61 4 61 4 61 4	63 1 61 3 58 8 60 6	63 1 61 3 58 8 60 6
7 58 7 58 7 58 7 58	18 48 20 46 23 41 21 43	2 64 4 62 7 57 5 59
59 6 59 6 59 6 59 6	63 1 61 3 58 8 60 6	47 17 45 19 42 24 44 22
64 1 64 1 64 1 64 1	58 8 60 6 63 1 61 3	58 8 60 6 63 1 61 3
4 61 4 61 4 61 4 61	7 57 5 59 2 64 4 62	7 57 5 59 2 64 4 62
58 7 58 7 58 7 58 7	42 24 44 22 47 17 45 19	58 8 60 6 63 1 61 3
6 59 6 59 6 59 6 59	7 57 5 59 2 64 4 62	23 41 21 43 18 48 20 46

```
 1/32/ 4/ 6/ 7/ 9/   ??    2/31/ 3/ 5/ 8/ 9/   ??    3/30/ 2/ 5/ 8/ 9/   ??
17 48 17 48 18 47 18 47   17 48 17 48 18 47 18 47   17 48 17 48 18 47 18 47
42 23 42 23 41 24 41 24   42 23 42 23 41 24 41 24   42 23 42 23 41 24 41 24
29 36 29 36 30 35 30 35   35 30 35 30 36 29 36 29   27 38 27 38 28 37 28 37
38 27 38 27 37 28 37 28   38 27 38 27 37 28 37 28   30 35 30 35 29 36 29 36
47 18 47 18 48 17 48 17   47 18 47 18 48 17 48 17   47 18 47 18 48 17 48 17
24 41 24 41 23 42 23 42   24 41 24 41 23 42 23 42   24 41 24 41 23 42 23 42
35 30 35 30 36 29 36 29   29 36 29 36 30 35 30 35   37 28 37 28 38 27 38 27
28 37 28 37 27 38 27 38   28 37 28 37 27 38 27 38   36 29 36 29 35 30 35 30

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

. . . . . .
```

But, what makes them wrong? How can we stop that?

We have to watch some examples of wrong answers above to analyze.

I noticed that we have to avoid using any combination of the same unit twice or more often, since it should make wrong answers against the **Def (3)** of Complete Euler Squares. I also noticed that using any combination of such pairs as `(1/, 32/)`, `(2/, 31/)`, `(3/, 30/)`, `(4/, 29/)`, `...`, `(15/, 18/)`, `(16/, 17/)` should be wrong. Each of those pairs is really a 'Self-Complementary' unit to the other partner.

I noticed that any combination of such pairs as `(1/, 2/)`, `(1/, 3/)`, `(1/, 5/)`, `(1/, 8/)`, `(1/, 31/)`, `(2/, 4/)`, `(2/, 7/)`, `...`, should also make the answer wrong.

I have finally known that the combination of any two units which are too similar or too different would surely make wrong answers against the Def (3) of C.Euler.S.

An idea came up to my mind. Why don't we measure how similar any two units are?

I counted how many digits are the same between any two units picked up and made the reference table of those data on memory array `mtc[33][33]` as shown below:

```
[Reference Table for the Best Match: 32 is the Best;]
 *|  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
--+-------------------------------------------------------------------------------------------------
 1| 64 48 48 32 32 32 32 32 48 32 32 16 32 32 32 32 32 32 32 32 48 32 32 16 32 32 32 32 32 16 16  0
 2| 48 64 32 48 32 32 32 32 32 48 16 32 32 32 32 32 32 32 32 32 32 48 16 32 32 32 32 32 16 32  0 16
 3| 48 32 64 48 32 32 32 32 32 16 48 32 32 32 32 32 32 32 32 32 32 16 48 32 32 32 32 32 16  0 32 16
 4| 32 48 48 64 32 32 32 32 16 32 32 48 32 32 32 32 32 32 32 32 32 16 32 32 48 32 32 32 32  0 16 16 32
 5| 32 32 32 32 64 48 48 32 32 32 32 32 48 32 32 16 48 32 32 16 32 32 32 32 32 16 16  0 32 32 32 32
 6| 32 32 32 32 48 64 32 48 32 32 32 32 32 48 16 32 32 48 16 32 32 32 32 32 16 32  0 16 32 32 32 32
 7| 32 32 32 32 48 32 64 48 32 32 32 32 32 16 48 32 32 16 48 32 32 32 32 32 16  0 32 16 32 32 32 32
 8| 32 32 32 32 32 48 48 64 32 32 32 32 16 32 32 48 16 32 32 48 32 32 32 32  0 16 16 32 32 32 32 32
 9| 48 32 32 16 32 32 32 32 64 48 48 32 32 32 32 32 32 32 32 32 32 32 16 16  0 32 32 32 32 48 32 32 16
10| 32 48 16 32 32 32 32 32 48 64 32 48 32 32 32 32 32 32 32 32 32 16 32  0 16 32 32 32 32 32 48 16 32
11| 32 16 48 32 32 32 32 32 48 32 64 48 32 32 32 32 32 32 32 32 32 16  0 32 16 32 32 32 32 32 16 48 32
12| 16 32 32 48 32 32 32 32 32 48 48 64 32 32 32 32 32 32 32 32  0 16 16 32 32 32 32 32 16 32 32 48
13| 32 32 32 32 48 32 32 16 32 32 32 32 64 48 48 32 32 16 16  0 32 32 32 32 48 32 32 16 32 32 32 32
14| 32 32 32 32 32 48 16 32 32 32 32 32 48 64 32 48 16 32  0 16 32 32 32 32 32 48 16 32 32 32 32 32
15| 32 32 32 32 32 16 48 32 32 32 32 32 48 32 64 48 16  0 32 16 32 32 32 32 32 16 48 32 32 32 32 32
16| 32 32 32 32 16 32 32 48 32 32 32 32 32 48 48 64  0 16 16 32 32 32 32 32 16 32 32 48 32 32 32 32
17| 32 32 32 32 48 32 32 16 32 32 32 32 32 16 16  0 64 48 48 32 32 32 32 32 48 32 32 16 32 32 32 32
18| 32 32 32 32 32 48 16 32 32 32 32 32 16 32  0 16 48 64 32 48 32 32 32 32 32 48 16 32 32 32 32 32
19| 32 32 32 32 32 16 48 32 32 32 32 32 16  0 32 16 48 32 64 48 32 32 32 32 32 16 48 32 32 32 32 32
20| 32 32 32 32 16 32 32 48 32 32 32 32  0 16 16 32 32 48 48 64 32 32 32 32 16 32 32 48 32 32 32 32
21| 48 32 32 16 32 32 32 32 32 16 16  0 32 32 32 32 32 32 32 32 64 48 48 32 32 32 32 32 48 32 32 16
22| 32 48 16 32 32 32 32 32 16 32  0 16 32 32 32 32 32 32 32 32 48 64 32 48 32 32 32 32 32 48 16 32
```

```
23| 32 16 48 32 32 32 32 32 16  0 32 16 32 32 32 32 32 32 32 32 48 32 64 48 32 32 32 32 32 16 48 32
24| 16 32 32 48 32 32 32 32  0 16 16 32 32 32 32 32 32 32 32 32 32 48 48 64 32 32 32 32 16 32 32 48
25| 32 32 32 32 32 16 16  0 32 32 32 32 48 32 32 16 48 32 32 16 32 32 32 32 64 48 48 32 32 32 32 32
26| 32 32 32 32 16 32  0 16 32 32 32 32 32 48 16 32 32 48 16 32 32 32 32 32 48 64 32 48 32 32 32 32
27| 32 32 32 32 16  0 32 16 32 32 32 32 32 16 48 32 32 16 48 32 32 32 32 32 48 32 64 48 32 32 32 32
28| 32 32 32 32  0 16 16 32 32 32 32 32 16 32 32 48 16 32 32 48 32 32 32 32 32 48 48 64 32 32 32 32
29| 32 16 16  0 32 32 32 32 48 32 32 16 32 32 32 32 32 32 32 32 48 32 32 16 32 32 32 32 64 48 48 32
30| 16 32  0 16 32 32 32 32 32 48 16 32 32 32 32 32 32 32 32 32 32 48 16 32 32 32 32 32 48 64 32 48
31| 16  0 32 16 32 32 32 32 32 16 48 32 32 32 32 32 32 32 32 32 32 16 48 32 32 32 32 32 48 32 64 48
32|  0 16 16 32 32 32 32 32 16 32 32 48 32 32 32 32 32 32 32 32 16 32 32 48 32 32 32 32 32 48 48 64
--+-----------------------------------------------------------------------------------------------
 *|  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
```

Why don't we consult with the above table before we combine any two Euler Units?

It is natural such pairs as `(1/, 1/)`, `(2/, 2/)`, `(3/, 3/)`, `(4/, 4/)`, `...`, `(31/, 31/)`, `(32/, 32/)` count 64. It means they are completely similar, since I compared the same unit to itself. If you use this combination, you will surely make your answer wrong against the `Def (3)`. Such pairs as `(1/, 32/)`, `(2/, 31/)`, `(3/, 30/)`, `...`, `(16/, 17/)`, `(17/, 16/)`, `...` count 0. This means they are completely different. If you use these 'complementary unit' pairs, you will be always wrong against the **`Def (3)`**.

On top of that I found if you use any pair which counts 16 or 48, you would be always wrong. Only when you use any pair which counts **32**, you would possibly be correct.

You should always consult with this reference table, just before you combine any two units, and you should choose only what counts **32**.

But you still need the check program for the **`Def (3)`** of 'C.E.S.', after combining and calculating under the next equation, before listing out all of your result.

```
Vn= An*32 + Bn*16 + Cn*8 + Dn*4 + En*2 + Fn*1 + 1;
```

Read the program list above carefully, and know what I have just wanted to do.

You also have to use the inequality conditions for your smart listing. One eighth of correct solutions can be listed out finally as the set of 'Standard Solutions' after eliminating any kind of imitations.

4-3. The Result of my recent Calculation

Here you are at the list of execution result of my program for the reconstruction of 'C&C' Pan-magic Squares of Order 8:

The result shows we have succeeded in composing 'Composite & Complete' PMS88, since the total count of solutions becomes the same with the one we got before.

It means that every 'C&C' MS88 is always 'Complete Euler Square' of order 8, and →

```
['Composite & Complete' Pan-magic Squares of Order 8 Reconstructed:
    List of Standard Solutions(n1==1): Used_Units////// Sol_No#|Sum_Checks|]
      1/       4/       5/       8/      10/      14/                      1#|Row|Clm\Pd1/Pd2|
01010101 01010101 01011010 01101001 01010101 00001111  1 63  5 59 14 52 10 56|260|260\260/260|
10101010 10101010 10100101 10010110 01010101 11110000 62  4 58  8 49 15 53 11|260|260\260/260|
01010101 10101010 01011010 01101001 01010101 00001111 17 47 21 43 30 36 26 40|260|260\260/260|
10101010 01010101 10100101 10010110 01010101 11110000 46 20 42 24 33 31 37 27|260|260\260/260|
10101010 10101010 01011010 01101001 10101010 00001111 51 13 55  9 64  2 60  6|260|260\260/260|
01010101 01010101 10100101 10010110 10101010 11110000 16 50 12 54  3 61  7 57|260|260\260/260|
10101010 01010101 01011010 01101001 10101010 00001111 35 29 39 25 48 18 44 22|260|260\260/260|
01010101 10101010 10100101 10010110 10101010 11110000 32 34 28 38 19 45 23 41|260|260\260/260|
      1/       4/       5/       8/      10/      15/                      2#|Row|Clm\Pd1/Pd2|
01010101 01010101 01011010 01101001 01010101 00111100  1 63  6 60 14 52  9 55|260|260\260/260|
10101010 10101010 10100101 10010110 01010101 11000011 62  4 57  7 49 15 54 12|260|260\260/260|
01010101 10101010 01011010 01101001 01010101 00111100 17 47 22 44 30 36 25 39|260|260\260/260|
10101010 01010101 10100101 10010110 01010101 11000011 46 20 41 23 33 31 38 28|260|260\260/260|
10101010 10101010 01011010 01101001 10101010 00111100 51 13 56 10 64  2 59  5|260|260\260/260|
01010101 01010101 10100101 10010110 10101010 11000011 16 50 11 53  3 61  8 58|260|260\260/260|
10101010 01010101 01011010 01101001 10101010 00111100 35 29 40 26 48 18 43 21|260|260\260/260|
01010101 10101010 10100101 10010110 10101010 11000011 32 34 27 37 19 45 24 42|260|260\260/260|
```

1/	4/	5/	8/	11/	14/								3#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	01010101	00001111	1	63	5	59	14	52	10	56|260|260\260/260|
10101010	10101010	10100101	10010110	01010101	11110000	62	4	58	8	49	15	53	11|260|260\260/260|
01010101	10101010	01011010	01101001	10101010	00001111	19	45	23	41	32	34	28	38|260|260\260/260|
10101010	01010101	10100101	10010110	10101010	11110000	48	18	44	22	35	29	39	25|260|260\260/260|
10101010	10101010	01011010	01101001	10101010	00001111	51	13	55	9	64	2	60	6|260|260\260/260|
01010101	01010101	10100101	10010110	10101010	11110000	16	50	12	54	3	61	7	57|260|260\260/260|
10101010	01010101	01011010	01101001	01010101	00001111	33	31	37	27	46	20	42	24|260|260\260/260|
01010101	10101010	10100101	10010110	01010101	11110000	30	36	26	40	17	47	21	43|260|260\260/260|

1/	4/	5/	8/	11/	15/								4#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	01010101	00111100	1	63	6	60	14	52	9	55|260|260\260/260|
10101010	10101010	10100101	10010110	01010101	11000011	62	4	57	7	49	15	54	12|260|260\260/260|
01010101	10101010	01011010	01101001	10101010	00111100	19	45	24	42	32	34	27	37|260|260\260/260|
10101010	01010101	10100101	10010110	10101010	11000011	48	18	43	21	35	29	40	26|260|260\260/260|
10101010	10101010	01011010	01101001	10101010	00111100	51	13	56	10	64	2	59	5|260|260\260/260|
01010101	01010101	10100101	10010110	10101010	11000011	16	50	11	53	3	61	8	58|260|260\260/260|
10101010	01010101	01011010	01101001	01010101	00111100	33	31	38	28	46	20	41	23|260|260\260/260|
01010101	10101010	10100101	10010110	01010101	11000011	30	36	25	39	17	47	22	44|260|260\260/260|

1/	4/	5/	8/	14/	10/								5#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	00001111	01010101	1	62	5	58	15	52	11	56|260|260\260/260|
10101010	10101010	10100101	10010110	11110000	01010101	63	4	59	8	49	14	53	10|260|260\260/260|
01010101	10101010	01011010	01101001	00001111	01010101	17	46	21	42	31	36	27	40|260|260\260/260|
10101010	01010101	10100101	10010110	11110000	01010101	47	20	43	24	33	30	37	26|260|260\260/260|
10101010	10101010	01011010	01101001	00001111	10101010	50	13	54	9	64	3	60	7|260|260\260/260|
01010101	01010101	10100101	10010110	11110000	10101010	16	51	12	55	2	61	6	57|260|260\260/260|
10101010	01010101	01011010	01101001	00001111	10101010	34	29	38	25	48	19	44	23|260|260\260/260|
01010101	10101010	10100101	10010110	11110000	10101010	32	35	28	39	18	45	22	41|260|260\260/260|

1/	4/	5/	8/	14/	11/								6#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	00001111	01010101	1	62	5	58	15	52	11	56|260|260\260/260|
10101010	10101010	10100101	10010110	11110000	01010101	63	4	59	8	49	14	53	10|260|260\260/260|
01010101	10101010	01011010	01101001	00001111	10101010	18	45	22	41	32	35	28	39|260|260\260/260|
10101010	01010101	10100101	10010110	11110000	10101010	48	19	44	23	34	29	38	25|260|260\260/260|
10101010	10101010	01011010	01101001	00001111	10101010	50	13	54	9	64	3	60	7|260|260\260/260|
01010101	01010101	10100101	10010110	11110000	10101010	16	51	12	55	2	61	6	57|260|260\260/260|
10101010	01010101	01011010	01101001	00001111	01010101	33	30	37	26	47	20	43	24|260|260\260/260|
01010101	10101010	10100101	10010110	11110000	01010101	31	36	27	40	17	46	21	42|260|260\260/260|

1/	4/	5/	8/	15/	10/								7#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	00111100	01010101	1	62	7	60	15	52	9	54|260|260\260/260|
10101010	10101010	10100101	10010110	11000011	01010101	63	4	57	6	49	14	55	12|260|260\260/260|
01010101	10101010	01011010	01101001	00111100	01010101	17	46	23	44	31	36	25	38|260|260\260/260|
10101010	01010101	10100101	10010110	11000011	01010101	47	20	41	22	33	30	39	28|260|260\260/260|
10101010	10101010	01011010	01101001	00111100	10101010	50	13	56	11	64	3	58	5|260|260\260/260|
01010101	01010101	10100101	10010110	11000011	10101010	16	51	10	53	2	61	8	59|260|260\260/260|
10101010	01010101	01011010	01101001	00111100	10101010	34	29	40	27	48	19	42	21|260|260\260/260|
01010101	10101010	10100101	10010110	11000011	10101010	32	35	26	37	18	45	24	43|260|260\260/260|

1/	4/	5/	8/	15/	11/								8#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01101001	00111100	01010101	1	62	7	60	15	52	9	54|260|260\260/260|
10101010	10101010	10100101	10010110	11000011	01010101	63	4	57	6	49	14	55	12|260|260\260/260|
01010101	10101010	01011010	01101001	00111100	10101010	18	45	24	43	32	35	26	37|260|260\260/260|
10101010	01010101	10100101	10010110	11000011	10101010	48	19	42	21	34	29	40	27|260|260\260/260|
10101010	10101010	01011010	01101001	00111100	10101010	50	13	56	11	64	3	58	5|260|260\260/260|
01010101	01010101	10100101	10010110	11000011	10101010	16	51	10	53	2	61	8	59|260|260\260/260|
10101010	01010101	01011010	01101001	00111100	01010101	33	30	39	28	47	20	41	22|260|260\260/260|
01010101	10101010	10100101	10010110	11000011	01010101	31	36	25	38	17	46	23	44|260|260\260/260|

1/	4/	5/	10/	8/	14/								9#|Row|Clm\Pd1/Pd2|
01010101	01010101	01011010	01010101	01101001	00001111	1	63	3	61	12	54	10	56|260|260\260/260|
10101010	10101010	10100101	01010101	10010110	11110000	60	6	58	8	49	15	51	13|260|260\260/260|
01010101	10101010	01011010	01010101	01101001	00001111	17	47	19	45	28	38	26	40|260|260\260/260|
10101010	01010101	10100101	01010101	10010110	11110000	44	22	42	24	33	31	35	29|260|260\260/260|
10101010	10101010	01011010	10101010	01101001	00001111	53	11	55	9	64	2	62	4|260|260\260/260|
01010101	01010101	10100101	10101010	10010110	11110000	16	50	14	52	5	59	7	57|260|260\260/260|
10101010	01010101	01011010	10101010	01101001	00001111	37	27	39	25	48	18	46	20|260|260\260/260|
01010101	10101010	10100101	10101010	10010110	11110000	32	34	30	36	21	43	23	41|260|260\260/260|

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

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

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

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

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

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

1/ 4/ 6/16/11/13/    70#
 1 59  5 52 14 56 10 63
62  8 58 15 49 11 53  4
19 41 23 34 32 38 28 45
48 22 44 29 35 25 39 18
51  9 55  2 64  6 60 13
16 54 12 61  3 57  7 50
33 27 37 20 46 24 42 31
30 40 26 47 17 43 21 36

1/ 4/ 7/ 6/16/11/    80#
 1 62 11 58 15 52  5 56
63  4 53  8 49 14 59 10
18 45 28 41 32 35 22 39
48 19 38 23 34 29 44 25
50 13 60  9 64  3 54  7
16 51  6 55  2 61 12 57
33 30 43 26 47 20 37 24
31 36 21 40 17 46 27 42

1/ 4/ 7/11/13/16/    90#
 1 61 10 63 12 56  3 54
60  8 51  6 49 13 58 15
21 41 30 43 32 36 23 34
48 20 39 18 37 25 46 27
53  9 62 11 64  4 55  2
16 52  7 50  5 57 14 59
33 29 42 31 44 24 35 22
28 40 19 38 17 45 26 47

1/ 4/ 7/13/16/11/   100#
 1 58 11 62 15 56  5 52
63  8 53  4 49 10 59 14
18 41 28 45 32 39 22 35
48 23 38 19 34 25 44 29
50  9 60 13 64  7 54  3
16 55  6 51  2 57 12 61
33 26 43 30 47 24 37 20
31 40 21 36 17 42 27 46

1/ 4/11/ 6/ 7/16/   200#
 1 63  4 59  8 58  5 62
56 10 53 14 49 15 52 11
25 39 28 35 32 34 29 38
48 18 45 22 41 23 44 19
57  7 60  3 64  2 61  6
16 50 13 54  9 55 12 51
33 31 36 27 40 26 37 30
24 42 21 46 17 47 20 43

1/ 4/14/11/ 5/15/   300#
 1 55  2 56 12 62 11 61
60 14 59 13 49  7 50  8
21 35 22 36 32 42 31 41
48 26 47 25 37 19 38 20
53  3 54  4 64 10 63  9
16 58 15 57  5 51  6 52
33 23 34 24 44 30 43 29
28 46 27 45 17 39 18 40

1/ 5/ 4/11/ 8/15/   400#
 1 63  4 62 20 46 17 47
60  6 57  7 41 23 44 22
13 51 16 50 32 34 29 35
56 10 53 11 37 27 40 26
45 19 48 18 64  2 61  3
24 42 21 43  5 59  8 58
33 31 36 30 52 14 49 15
28 38 25 39  9 55 12 54

1/ 5/11/ 8/ 4/15/   500#
 1 63  6 60 22 44 17 47
56 10 51 13 35 29 40 26
11 53 16 50 32 34 27 37
62  4 57  7 41 23 46 20
43 21 48 18 64  2 59  5
30 36 25 39  9 55 14 52
33 31 38 28 54 12 49 15
24 42 19 45  3 61  8 58

1/ 5/15/14/11/10/   600#
 1 52  9 60 29 48 21 40
61 16 53  8 33 20 41 28
 3 50 11 58 31 46 23 38
63 14 55  6 35 18 43 26
36 17 44 25 64 13 56  5
32 45 24 37  4 49 12 57
34 19 42 27 62 15 54  7
30 47 22 39  2 51 10 59

1/ 6/10/13/16/11/   700#
 1 58  3 46 23 48 21 60
55 16 53 28 33 26 35 14
 2 57  4 45 24 47 22 59
56 15 54 27 34 25 36 13
42 17 44  5 64  7 62 19
32 39 30 51 10 49 12 37
41 18 43  6 63  8 61 20
31 40 29 52  9 50 11 38

1/ 6/16/10/11/13/   800#
 1 55  9 40 26 48 18 63
58 16 50 31 33 23 41  8
 3 53 11 38 28 46 20 61
60 14 52 29 35 21 43  6
39 17 47  2 64 10 56 25
32 42 24 57  7 49 15 34
37 19 45  4 62 12 54 27
30 44 22 59  5 51 13 36

1/ 7/10/ 6/13/11/   900#
 1 62 17 60 23 44  7 46
55 12 39 14 33 30 49 28
 2 61 18 59 24 43  8 45
56 11 40 13 34 29 50 27
42 21 58 19 64  3 48  5
32 35 16 37 10 53 26 51
41 22 57 20 63  4 47  6
31 36 15 38  9 54 25 52

1/ 7/16/ 4/10/13/  1000#
 1 55 25 56 26 48  2 47
62 12 38 11 37 19 61 20
 5 51 29 52 30 44  6 43
58 16 34 15 33 23 57 24
39 17 63 18 64 10 40  9
28 46  4 45  3 53 27 54
35 21 59 22 60 14 36 13
32 42  8 41  7 49 31 50

1/11/16/ 7/10/13/  2000#
 1 55 13 56 14 60  2 59
46 28 34 27 33 23 45 24
17 39 29 40 30 44 18 43
62 12 50 11 49  7 61  8
51  5 63  6 64 10 52  9
32 42 20 41 19 37 31 38
35 21 47 22 48 26 36 25
16 58  4 57  3 53 15 54

2/ 3/ 8/ 9/14/15/  3000#
 1 61 10 54 12 56  3 63
60  8 51 15 49 13 58  6
17 45 26 38 28 40 19 47
32 36 23 43 21 41 30 34
53  9 62  2 64  4 55 11
16 52  7 59  5 57 14 50
37 25 46 18 48 20 39 27
44 24 35 31 33 29 42 22
```

```
 2/12/14/  8/ 3/15/   4000#    3/ 6/13/ 9/ 2/16/   5000#    3/12/ 7/ 6/ 2/16/   6000#
  1 55  6 52 14 60  9 63      1 55  2 47 26 48 25 56      1 63 10 59 14 52  5 56
 48 26 43 29 35 21 40 18     60 14 59 22 35 21 36 13     48 18 39 22 35 29 44 25
 19 37 24 34 32 42 27 45     33 23 34 15 58 16 57 24     49 15 58 11 62  4 53  8
 16 58 11 61  3 53  8 50     62 12 61 20 37 19 38 11     46 20 37 24 33 31 42 27
 51  5 56  2 64 10 59 13     39 17 40  9 64 10 63 18     51 13 60  9 64  2 55  6
 30 44 25 47 17 39 22 36     30 44 29 52  5 51  6 43     30 36 21 40 17 47 26 43
 33 23 38 20 46 28 41 31      7 49  8 41 32 42 31 50      3 61 12 57 16 50  7 54
 62 12 57 15 49  7 54  4     28 46 27 54  3 53  4 45     32 34 23 38 19 45 28 41

 3/16/ 7/12/ 2/13/   7000#    4/11/ 6/16/13/10/   8000#    6/ 3/16/ 9/ 2/13/   9000#
  1 47 25 48 26 56  2 55      1 58  5 52 15 56 11 62      1 55  9 24 42 32 34 63
 60 22 36 21 35 13 59 14     47 24 43 30 33 26 37 20     60 14 52 45 19 37 27  6
 37 11 61 12 62 20 38 19     49 10 53  4 63  8 59 14     17 39 25  8 58 16 50 47
 58 24 34 23 33 15 57 16     31 40 27 46 17 42 21 36     62 12 54 43 21 35 29  4
 39  9 63 10 64 18 40 17     50  9 54  3 64  7 60 13     23 33 31  2 64 10 56 41
 30 52  6 51  5 43 29 44     32 39 28 45 18 41 22 35     46 28 38 59  5 51 13 20
  3 45 27 46 28 54  4 53      2 57  6 51 16 55 12 61      7 49 15 18 48 26 40 57
 32 50  8 49  7 41 31 42     48 23 44 29 34 25 38 19     44 30 36 61  3 53 11 22

 6/16/12/ 9/13/ 3/ 10000#    8/11/ 1/14/15/10/ 11000#    9/ 3/ 6/ 7/13/12/ 12000#
  1 46 17 16 51 32 35 62      1 58 35 28 39 32  5 62      1 62  5 56 15 52 11 58
 52 31 36 61  2 45 18 15     47 24 13 54  9 50 43 20     31 36 27 42 17 46 21 40
 10 37 26  7 60 23 44 53     17 42 51 12 55 16 21 46     18 45 22 39 32 35 28 41
 56 27 40 57  6 41 22 11     63  8 29 38 25 34 59  4     63  4 59 10 49 14 53  8
 14 33 30  3 64 19 48 49     26 33 60  3 64  7 30 37     50 13 54  7 64  3 60  9
 63 20 47 50 13 34 29  4     56 15 22 45 18 41 52 11     48 19 44 25 34 29 38 23
  5 42 21 12 55 28 39 58     10 49 44 19 48 23 14 53     33 30 37 24 47 20 43 26
 59 24 43 54  9 38 25  8     40 31  6 61  2 57 36 27     16 51 12 57  2 61  6 55

 9/ 8/ 3/14/15/12/ 13000#    9/15/ 2/14/ 5/12/ 14000#   10/ 6/11/ 1/13/16/ 15000#
  1 58 19 44 23 48  5 62      1 44 17 60 23 62  7 46      1 61  2 47 20 48 19 62
 31 40 13 54  9 50 27 36     31 54 15 38  9 36 25 52     24 44 23 58  5 57  6 43
 10 49 28 35 32 39 14 53      2 43 18 59 24 61  8 45      9 53 10 39 28 40 27 54
 63  8 45 22 41 18 59  4     55 30 39 14 33 12 49 28     32 36 31 50 13 49 14 35
 42 17 60  3 64  7 46 21     42  3 58 19 64 21 48  5     45 17 46  3 64  4 63 18
 56 15 38 29 34 25 52 11     56 29 40 13 34 11 50 27     60  8 59 22 41 21 42  7
 33 26 51 12 55 16 37 30     41  4 57 20 63 22 47  6     37 25 38 11 56 12 55 26
 24 47  6 61  2 57 20 43     32 53 16 37 10 35 26 51     52 16 51 30 33 29 34 15

11/ 1/10/16/13/ 7/ 16000#   11/ 7/ 4/ 1/13/16/ 17000#   11/14/ 1/ 5/ 4/15/ 18000#
  1 58  6 60  8 63  3 61      1 61 18 63 20 48  3 46      1 47  2 48 22 60 21 59
 24 47 19 45 17 42 22 44     32 36 15 34 13 49 30 51     32 50 31 49 11 37 12 38
 33 26 38 28 40 31 35 29     41 21 58 23 60  8 43  6     35 13 36 14 56 26 55 25
 56 15 51 13 49 10 54 12     56 12 39 10 37 25 54 27     62 20 61 19 41  7 42  8
 57  2 62  4 64  7 59  5     45 17 62 19 64  4 47  2     43  5 44  6 64 18 63 17
 48 23 43 21 41 18 46 20     52 16 35 14 33 29 50 31     54 28 53 27 33 15 34 16
 25 34 30 36 32 39 27 37      5 57 22 59 24 44  7 42      9 39 10 40 30 52 29 51
 16 55 11 53  9 50 14 52     28 40 11 38  9 53 26 55     24 58 23 57  3 45  4 46

12/ 3/ 9/ 6/ 7/16/ 19000#   12/16/ 9/13/ 7/ 3/ 20000#   14/ 9/ 8/ 5/ 3/12/ 21000#
  1 63  4 59  8 58  5 62      1 44 19 48 23 62  5 58      1 32  9 24 45 52 37 60
 24 42 21 46 17 47 20 43     24 61  6 57  2 43 20 47     47 50 39 58  3 30 11 22
 49 15 52 11 56 10 53 14     34 11 52 15 56 29 38 25      4 29 12 21 48 49 40 57
 32 34 29 38 25 39 28 35     32 53 14 49 10 35 28 39     63 34 55 42 19 14 27  6
 57  7 60  3 64  2 61  6     42  3 60  7 64 21 46 17     20 13 28  5 64 33 56 41
 48 18 45 22 41 23 44 19     63 22 45 18 41  4 59  8     62 35 54 43 18 15 26  7
  9 55 12 51 16 50 13 54      9 36 27 40 31 54 13 50     17 16 25  8 61 36 53 44
 40 26 37 30 33 31 36 27     55 30 37 26 33 12 51 16     46 51 38 59  2 31 10 23

16/ 3/12/ 6/ 9/13/ 22000#   16/12/ 6/ 9/ 3/13/ 23000#
  1 31 33 28 38 60  6 63      1 31 33 24 42 56 10 63
 54 44 22 47 17 15 49 12     44 54 12 61  3 29 35 22
 25  7 57  4 62 36 30 39     19 13 51  6 60 38 28 45
 56 42 24 45 19 13 51 10     48 50 16 57  7 25 39 18
 27  5 59  2 64 34 32 37     23  9 55  2 64 34 32 41
 48 50 16 53 11 21 43 18     62 36 30 43 21 11 53  4
  3 29 35 26 40 58  8 61      5 27 37 20 46 52 14 59
 46 52 14 55  9 23 41 20     58 40 26 47 17 15 49  8    . . . . .
```

```
 [Solution Counts: n1==1/Total = 23040/368640]

 [Classify the Counts according to the Value of n1]
 [ 1] 23040,  [ 2] 23040,  [ 3] 22656,  [ 4] 22656,  [ 5] 21696,  [ 6] 21696,
 [ 7] 21312,  [ 8] 21312,  [ 9] 18240,  [10] 18240,  [11] 17856,  [12] 17856,
 [13] 16896,  [14] 16896,  [15] 16512,  [16] 16512,  [17]  6528,  [18]  6528,
 [19]  6144,  [20]  6144,  [21]  5184,  [22]  5184,  [23]  4800,  [24]  4800,
 [25]  1728,  [26]  1728,  [27]  1344,  [28]  1344,  [29]   384,  [30]   384,
 [31]     0,  [32]     0,  [33]     0,  [34]     0,  [35]     0,  [36]     0,
 . . . . .
 [OK!]
** Calculated and Listed by Kanji Setsuda
    on May 14, 2012 with MacOSX 10.7.4 & Xcode 4.3.2 **
```

→ the solution set of 'C&C' MS88 is included in the set of 'Complete Euler Squares' of Order 8. The former makes the subset of the latter.

I was extremely surprised when I felt the execution **speed** of this program. It took only a few minutes even for my PC to list them up to the last. How wonderful it was!

5. How to Get the Complete List of 90 Fundamental Solutions

It is easy for you to get the list of 90 Fundamental Solutions of 'C&C' MS88 with n1=1. You may only add the selection program to the one above.

But if you want to have such a faster program as you can, you have to cut off the waste part of the program above.

Since the fundamental solutions are picked up from the solution set of standard form with n1= 1 in general, you could cut off half of Latin Squares to get only 16.

```
A1*32 + B1*16 + C1*8 + D1*4 + E1*2 + F1*1 + 1 = 1
Therefore A1=0; B1=0; C1=0; D1=0; E1=0; F1=0
```

And add the next inequality conditions complex before listing.

```
//
    if((nm[1]<nm[64])&&(nm[1]<nm[3])&&(nm[3]<nm[7])&&(nm[7]<nm[5])){
      if((nm[1]<nm[8])&&(nm[2]>nm[4])&&(nm[4]>nm[8])&&(nm[8]>nm[6])){
        if((nm[1]<nm[57])&&(nm[1]<nm[17])&&(nm[17]<nm[49])&&(nm[49]<nm[33])){
          if((nm[2]>nm[9])&&(nm[9]>nm[25])&&(nm[25]>nm[57])&&(nm[57]>nm[41])){solrecord();}
        }}}
      ......
//
```

The next list shows the result of my recent calculation.

```
** 'Composite & Complete' Pan-magic Squares of Order 8:  **
**  List of the 90 Fundamental Solutions with n1=1 **

 [List of Euler Units of Binary Decompositions]
      1/       2/       3/       4/       5/       6/       7/       8/
 01010101 01010101 01010101 01010101 01011010 01001011 01111000 01101001
 10101010 10101010 10101010 10101010 10100101 10110100 10000111 10010110
 01010101 01010101 10101010 10101010 01011010 01001011 01111000 01101001
 10101010 01010101 10101010 01010101 10100101 10110100 10000111 10010110
 10101010 10101010 10101010 10101010 01011010 01001011 01111000 01101001
 01010101 01010101 01010101 01010101 10100101 10110100 10000111 10010110
 10101010 10101010 01010101 01010101 01011010 01001011 01111000 01101001
 01010101 10101010 01010101 10101010 10100101 10110100 10000111 10010110
      9/      10/      11/      12/      13/      14/      15/      16/
 01010101 01010101 01010101 01010101 00011110 00001111 00111100 00101101
 01010101 01010101 01010101 01010101 11100001 11110000 11000011 11010010
 01010101 01010101 10101010 10101010 00011110 00001111 00111100 00101101
 10101010 01010101 10101010 01010101 11100001 11110000 11000011 11010010
```

```
10101010  10101010  10101010  10101010  00011110  00001111  00111100  00101101
10101010  10101010  10101010  10101010  11100001  11110000  11000011  11010010
10101010  10101010  01010101  01010101  00011110  00001111  00111100  00101101
01010101  10101010  01010101  10101010  11100001  11110000  11000011  11010010
```

[Count of Latin Units = 16]

['Composite & Complete' Pan-magic Squares of Order 8 Reconstructed:
Smart List of the **90 Fundamental Solutions**: Used_Units////// Sol_Number#]

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

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

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

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

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

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

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

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

  5/ 1/ 4/11/14/ 8/  70#      5/ 1/ 8/10/15/ 4/  80#      5/ 1/10/14/ 8/ 4/  90#
 1 62  2 61 36 31 35 32      1 62 11 56 43 24 33 30      1 60  3 58 39 30 37 32
60  7 59  8 25 38 26 37     60  7 50 13 18 45 28 39     56 13 54 15 18 43 20 41
13 50 14 49 48 19 47 20      2 61 12 55 44 23 34 29      2 59  4 57 40 29 38 31
56 11 55 12 21 42 22 41     59  8 49 14 17 46 27 40     55 14 53 16 17 44 19 42
29 34 30 33 64  3 63  4     22 41 32 35 64  3 54  9     26 35 28 33 64  5 62  7
40 27 39 28  5 58  6 57     47 20 37 26  5 58 15 52     47 22 45 24  9 52 11 50
17 46 18 45 52 15 51 16     21 42 31 36 63  4 53 10     25 36 27 34 63  6 61  8
44 23 43 24  9 54 10 53     48 19 38 25  6 57 16 51     48 21 46 23 10 51 12 49
[Solution Counts(n1=1) = 90]  [OK!]

** Monitor List of Solution Correspondences between Old List and New **
 ??: 0
  1: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 31: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 61: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [OK!]
```

This monitor list tells us we have surely succeeded in getting the complete set of 90 Fundamental Solutions. Neither repetition nor drop-off of any solution could be found.

6. How to Compose 'Composite and Pan-diagonal' Magic Squares of Order 8

[Basic Positions]

```
61 62 63 64 57 58 59 60 61 62 63 64 57 58 59 60
           .-----------------------.
n5 n6 n7 n8|n1|n2|n3|n4|n5|n6|n7|n8|n1 n2 n3 n4
           |--+--+--+--+--+--+--+--|
13 14 15 16|n9|10|11|12|13|14|15|16|n9 10 11 12
           |--+--+--+--+--+--+--+--|
21 22 23 24|17|18|19|20|21|22|23|24|17 18 19 20
           |--+--+--+--+--+--+--+--|
29 30 31 32|25|26|27|28|29|30|31|32|25 26 27 28
           |--+--+--+--+--+--+--+--|
37 38 39 40|33|34|35|36|37|38|39|40|33 34 35 36
           |--+--+--+--+--+--+--+--|
45 46 47 48|41|42|43|44|45|46|47|48|41 42 43 44
           |--+--+--+--+--+--+--+--|
53 54 55 56|49|50|51|52|53|54|55|56|49 50 51 52
           |--+--+--+--+--+--+--+--|
61 62 63 64|57|58|59|60|61|62|63|64|57 58 59 60
           '-----------------------'
n5 n6 n7 n8 n1 n2 n3 n4 n5 n6 n7 n8 n1 n2 n3 n4
```

**** 'Composite Conditions': ****

```
n1+n2+n9+n10=2;
n2+n3+n10+n11=2;
n3+n4+n11+n12=2;
n4+n5+n12+n13=2;
n5+n6+n13+n14=2;
n6+n7+n14+n15=2;
n7+n8+n15+n16=2;
n8+n1+n16+n9=2;
n9+n10+n17+n18=2;
n10+n11+n18+n19=2;
n11+n12+n19+n20=2;
n12+n13+n20+n21=2;
n13+n14+n21+n22=2;
n14+n15+n22+n23=2;
n15+n16+n23+n24=2;
n16+n9+n24+n17=2;
n17+n18+n25+n26=2;
n18+n19+n26+n27=2;
n19+n20+n27+n28=2;
n20+n21+n28+n29=2;
```

```
    n21+n22+n29+n30=2;   n22+n23+n30+n31=2;   n23+n24+n31+n32=2;
    n24+n17+n32+n25=2;   n25+n26+n33+n34=2;   n26+n27+n34+n35=2;
    n27+n28+n35+n36=2;   n28+n29+n36+n37=2;   n29+n30+n37+n38=2;
    n30+n31+n38+n39=2;   n31+n32+n39+n40=2;   n32+n25+n40+n33=2;   . . . . . .

  ** Conditions for every Row, Column and Pan-diagonal **
n1+n2+n3+n4+n5+n6+n7+n8=4;          n1+n9+n17+n25+n33+n41+n49+n57=4;
n9+n10+n11+n12+n13+n14+n15+n16=4;   n2+n10+n18+n26+n34+n42+n50+n58=4;
n17+n18+n19+n20+n21+n22+n23+n24=4;  n3+n11+n19+n27+n35+n43+n51+n59=4;
n25+n26+n27+n28+n29+n30+n31+n32=4;  n4+n12+n20+n28+n36+n44+n52+n60=4;
n33+n34+n35+n36+n37+n38+n39+n40=4;  n5+n13+n21+n29+n37+n45+n53+n61=4;
n41+n42+n43+n44+n45+n46+n47+n48=4;  n6+n14+n22+n30+n38+n46+n54+n62=4;
n49+n50+n51+n52+n53+n54+n55+n56=4;  n7+n15+n23+n31+n39+n47+n55+n63=4;
n57+n58+n59+n60+n61+n62+n63+n64=4;  n8+n16+n24+n32+n40+n48+n56+n64=4;

  * Pan-diagonal Conditions: *
n1+n10+n19+n28+n37+n46+n55+n64=4;   n1+n16+n23+n30+n37+n44+n51+n58=4;
n2+n11+n20+n29+n38+n47+n56+n57=4;   n2+n9+n24+n31+n38+n45+n52+n59=4;
n3+n12+n21+n30+n39+n48+n49+n58=4;   n3+n10+n17+n32+n39+n46+n53+n60=4;
n4+n13+n22+n31+n40+n41+n50+n59=4;   n4+n11+n18+n25+n40+n47+n54+n61=4;
n5+n14+n23+n32+n33+n42+n51+n60=4;   n5+n12+n19+n26+n33+n48+n55+n62=4;
n6+n15+n24+n25+n34+n43+n52+n61=4;   n6+n13+n20+n27+n34+n41+n56+n63=4;
n7+n16+n17+n26+n35+n44+n53+n62=4;   n7+n14+n21+n28+n35+n42+n49+n64=4;
n8+n9+n18+n27+n36+n45+n54+n63=4;    n8+n15+n22+n29+n36+n43+n50+n57=4;
```

For the next case, let's study about composing 'Composite and Pan-diagonal' Magic Squares of Order 8 by our New Euler's Method with Binary Number System, shall we?

I usually use those two words 'Complete' and 'Pan-diagonal' in different meanings. Basic equations of definition are partially different from each other.

'Pan-diagonal' type has to be composed only under the basic conditions above, while 'Complete' one has to be made with both the next 'Complete Conditions' and the basic ones above at the same time.

```
** [Complete Conditions](We don't use this time) **
  ** Complementary Pairs must be located as follows: **
  n1+n37=1;   n2+n38=1;   n3+n39=1;   n4+n40=1;   n5+n33=1;
  n6+n34=1;   n7+n35=1;   n8+n36=1;   n9+n45=1;   n10+n46=1;
  n11+n47=1;  n12+n48=1;  n13+n41=1;  n14+n42=1;  n15+n43=1;
  n16+n44=1;  n17+n53=1;  n18+n54=1;  n19+n55=1;  n20+n56=1;
  n21+n49=1;  n22+n50=1;  n23+n51=1;  n24+n52=1;  n25+n61=1;
  n26+n62=1;  n27+n63=1;  n28+n64=1;  n29+n57=1;  n30+n58=1;
  n31+n59=1;  n32+n60=1;
```

Now that we are going to make the 'Composite & Pan-diagonal' MS88, we don't take these 'Complete Conditions' here. Therefore it is supposed that we will have the larger set of solutions than the one of 'Composite & Complete' MS88.

6-1. Make Complete Euler Squares of Order 8 for that type

Let me show you my newest program for 'C&P' MS88 and its execution results.

As I defined every value of n1~n64 individually in each procedure, I had to write a simple but a long text.

```
//** 'Composite & Pan-diagonal' Magic Squares of Order 8:  **
//**  Composing 'Complete Euler Squares' of Order 8 by the **
//**  Binary Number System and 'New Euler's Method' **
//** 'CES88CPD.c': Dictated by Kanji Setsda in 2003, 2006, 2011; **
//**  Revised on May 13, 2012 with MacOSX 10.7.4 and Xcode 4.3.2 **
//
#include <stdio.h>
//
long int cnt, cnt1, cnt2;
```

```
long cntr[4];
long n1c[65];
short LSM, lcnt, cnt3;
short nm[65], uflg[65];
short mtc[73][73];
short tlu[73][65];
short anm[4][72];
//
short bs[4][2], cs[65][2];
//
short u1,u2,u3,u4,u5,u6;
//
void stp01(void), stp02(void), stp03(void), stp04(void);
void stp05(void), stp06(void), stp07(void), stp08(void);
void stp09(void), stp10(void), stp11(void), stp12(void);
void stp13(void), stp14(void), stp15(void), stp16(void);
void stp17(void), stp18(void), stp19(void), stp20(void);
void stp21(void), stp22(void), stp23(void), stp24(void);
void stp25(void), stp26(void), stp27(void), stp28(void);
void stp29(void), stp30(void), stp31(void), stp32(void);
void stp33(void), stp34(void), stp35(void), stp36(void);
void stp37(void), stp38(void), stp39(void), stp40(void);
void stp41(void), stp42(void), stp43(void), stp44(void);
void stp45(void), stp46(void), stp47(void), stp48(void);
void stp49(void), stp50(void), stp51(void), stp52(void);
void stp53(void), stp54(void), stp55(void), stp56(void);
void stp57(void), stp58(void), stp59(void), stp60(void);
void stp61(void), stp62(void), stp63(void), stp64(void);
void lurecord(void);
//
void prlunit(void);
void cmbcmp(void);
void cmcm2(void), cmcm3(void), cmcm4(void), cmcm5(void), cmcm6(void), cmcm7(void);
void solrecord(void);
void solprd(void);
void pr3sol(short x);
void prn1c(void);
//
//* Main Program *
int main(){
 short m,n;
  printf("\n");
  printf("** 'Composite & Pan-diagonal' Magic Squares of Order 8:  **\n");
  printf("**  Composing 'Complete Euler Squares' of Order 8 by the **\n");
  printf("**  Binary Number System and 'New Euler's Method'  **\n");
  for(m=0;m<4;m++){for(n=0;n<2;n++){bs[m][n]=0;}}
  for(m=0;m<65;m++){for(n=0;n<2;n++){cs[m][n]=0;}}
  lcnt=0;
  stp01();           //Make the Latin Units
  printf("\n");
  printf(" [List of Latin Units by the Binary Decompositions]\n");
  prlunit();         //Print the Latin Units
  printf("\n");
  printf(" [Compositions of 'Composite & Pan-diagonal' Magic Squares 8x8:\n");
  printf("   Solutions(n1==1) with Used_Units////// S_No#; Sum_Checks||\\/|]\n");
  LSM=260; cnt=0; cnt1=0; cnt3=0;
  for(n=0;n<65;n++){n1c[n]=0;}
  cmbcmp();          //Combine, Compose and Print */
  if(cnt3>0){pr3sol(cnt3);}
  printf("  . . . . .\n");
  printf(" [Solution Counts(n1==1)/Total = %ld/%ld]\n",cnt1,cnt);
  printf("\n");
  printf(" [List of Counts according to the Value of n1]\n");
  prn1c();
  printf("  . . . . .\n");
```

```
  printf(" [OK!]\n");
  printf("** Calculated and Listed by Kanji Setsuda **\n");
  printf("    on May 13, 2012 with MacOSX 10.7.4 & Xcode 4.3.2 **\n");
  printf("\n");
 return 0;
}
//
//** Program Modules for Latin Units of 'Complete Euler Squares' **
//** for the 'Composite & Pan-diagonal' Magic Squares of Order 8 **
//** by the Binary Decompositions and the 'New Euler's Method' **
//** Listed by Kanji Setsuda on May 13, 2013 with Mac Xcode 4 **
// Level #1:
// Set n1
void stp01(){
 short a;
  for(a=0;a<2;a++){
    if((bs[1][a]<4)&&(bs[2][a]<4)&&(bs[3][a]<4)){
      if((cs[1][a]<2)&&(cs[8][a]<2)&&(cs[57][a]<2)&&(cs[64][a]<2)){
        nm[1]=a;
        bs[1][a]++; bs[2][a]++; bs[3][a]++;
        cs[1][a]++; cs[8][a]++; cs[57][a]++; cs[64][a]++;
        stp02();
        cs[64][a]--; cs[57][a]--; cs[8][a]--; cs[1][a]--;
        bs[3][a]--; bs[2][a]--; bs[1][a]--;
      }}
  }
}
// Set n2
void stp02(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[1][a]<4){
      if((cs[1][a]<2)&&(cs[2][a]<2)&&(cs[57][a]<2)&&(cs[58][a]<2)){
        nm[2]=a;
        bs[1][a]++;
        cs[1][a]++; cs[2][a]++; cs[57][a]++; cs[58][a]++;
        stp03();
        cs[58][a]--; cs[57][a]--; cs[2][a]--; cs[1][a]--;
        bs[1][a]--;
      }}
  }
}
// Set n9
void stp03(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[2][a]<4){
      if((cs[1][a]<2)&&(cs[8][a]<2)&&(cs[9][a]<2)&&(cs[16][a]<2)){
        nm[9]=a;
        bs[2][a]++;
        cs[1][a]++; cs[8][a]++; cs[9][a]++; cs[16][a]++;
        stp04();
        cs[16][a]--; cs[9][a]--; cs[8][a]--; cs[1][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n10
void stp04(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[1][a]<2)&&(cs[2][a]<2)&&(cs[9][a]<2)&&(cs[10][a]<2)){
        nm[10]=a;
        bs[3][a]++;
```

```
          cs[1][a]++; cs[2][a]++; cs[9][a]++; cs[10][a]++;
          stp05();
          cs[10][a]--; cs[9][a]--; cs[2][a]--; cs[1][a]--;
          bs[3][a]--;
        }}
    }
}
// Set n3
void stp05(){
 short a;
  for(a=0;a<2;a++){
    if(bs[1][a]<4){
      if((cs[2][a]<2)&&(cs[3][a]<2)&&(cs[58][a]<2)&&(cs[59][a]<2)){
        nm[3]=a;
        bs[1][a]++;
        cs[2][a]++; cs[3][a]++; cs[58][a]++; cs[59][a]++;
        stp06();
        cs[59][a]--; cs[58][a]--; cs[3][a]--; cs[2][a]--;
        bs[1][a]--;
      }}
  }
}
// Set n11
void stp06(){
 short a;
  for(a=0;a<2;a++){
    if((cs[2][a]<2)&&(cs[3][a]<2)&&(cs[10][a]<2)&&(cs[11][a]<2)){
      nm[11]=a;
      cs[2][a]++; cs[3][a]++; cs[10][a]++; cs[11][a]++;
      stp07();
      cs[11][a]--; cs[10][a]--; cs[3][a]--; cs[2][a]--;
    }
  }
}
// Set n4
void stp07(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[1][a]<4){
      if((cs[3][a]<2)&&(cs[4][a]<2)&&(cs[59][a]<2)&&(cs[60][a]<2)){
        nm[4]=a;
        bs[1][a]++;
        cs[3][a]++; cs[4][a]++; cs[59][a]++; cs[60][a]++;
        stp08();
        cs[60][a]--; cs[59][a]--; cs[4][a]--; cs[3][a]--;
        bs[1][a]--;
      }}
  }
}
// Set n12
void stp08(){
 short a;
  for(a=0;a<2;a++){
    if((cs[3][a]<2)&&(cs[4][a]<2)&&(cs[11][a]<2)&&(cs[12][a]<2)){
      nm[12]=a;
      cs[3][a]++; cs[4][a]++; cs[11][a]++; cs[12][a]++;
      stp09();
      cs[12][a]--; cs[11][a]--; cs[4][a]--; cs[3][a]--;
    }
  }
}
// Set n5
void stp09(){
 short a;
  for(a=0;a<2;a++){
```

```
    if(bs[1][a]<4){
      if((cs[4][a]<2)&&(cs[5][a]<2)&&(cs[60][a]<2)&&(cs[61][a]<2)){
        nm[5]=a;
        bs[1][a]++;
        cs[4][a]++; cs[5][a]++; cs[60][a]++; cs[61][a]++;
        stp10();
        cs[61][a]--; cs[60][a]--; cs[5][a]--; cs[4][a]--;
        bs[1][a]--;
      }}
  }
}
// Set n13
void stp10(){
 short a;
  for(a=0;a<2;a++){
    if((cs[4][a]<2)&&(cs[5][a]<2)&&(cs[12][a]<2)&&(cs[13][a]<2)){
      nm[13]=a;
      cs[4][a]++; cs[5][a]++; cs[12][a]++; cs[13][a]++;
      stp11();
      cs[13][a]--; cs[12][a]--; cs[5][a]--; cs[4][a]--;
    }
  }
}
// Set n6
void stp11(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[1][a]<4){
      if((cs[5][a]<2)&&(cs[6][a]<2)&&(cs[61][a]<2)&&(cs[62][a]<2)){
        nm[6]=a;
        bs[1][a]++;
        cs[5][a]++; cs[6][a]++; cs[61][a]++; cs[62][a]++;
        stp12();
        cs[62][a]--; cs[61][a]--; cs[6][a]--; cs[5][a]--;
        bs[1][a]--;
      }}
  }
}
// Set n14
void stp12(){
 short a;
  for(a=0;a<2;a++){
    if((cs[5][a]<2)&&(cs[6][a]<2)&&(cs[13][a]<2)&&(cs[14][a]<2)){
      nm[14]=a;
      cs[5][a]++; cs[6][a]++; cs[13][a]++; cs[14][a]++;
      stp13();
      cs[14][a]--; cs[13][a]--; cs[6][a]--; cs[5][a]--;
    }
  }
}
// Set n7
void stp13(){
 short a;
  for(a=0;a<2;a++){
    if(bs[1][a]<4){
      if((cs[6][a]<2)&&(cs[7][a]<2)&&(cs[62][a]<2)&&(cs[63][a]<2)){
        nm[7]=a;
        bs[1][a]++;
        cs[6][a]++; cs[7][a]++; cs[62][a]++; cs[63][a]++;
        stp14();
        cs[63][a]--; cs[62][a]--; cs[7][a]--; cs[6][a]--;
        bs[1][a]--;
      }}
  }
}
```

```
// Set n15
void stp14(){
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[6][a]<2)&&(cs[7][a]<2)&&(cs[14][a]<2)&&(cs[15][a]<2)){
        nm[15]=a;
        bs[4][a]++;
        cs[6][a]++; cs[7][a]++; cs[14][a]++; cs[15][a]++;
        stp15();
        cs[15][a]--; cs[14][a]--; cs[7][a]--; cs[6][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n8
void stp15(){
 short a;
  for(a=1;a>=0;a--){
    if((bs[1][a]<4)&&(bs[4][a]<4)){
      if((cs[7][a]<2)&&(cs[8][a]<2)&&(cs[63][a]<2)&&(cs[64][a]<2)){
        nm[8]=a;
        bs[1][a]++; bs[4][a]++;
        cs[7][a]++; cs[8][a]++; cs[63][a]++; cs[64][a]++;
        stp16();
        cs[64][a]--; cs[63][a]--; cs[8][a]--; cs[7][a]--;
        bs[4][a]--; bs[1][a]--;
      }}
  }
}
// Set n16
void stp16(){
 short a;
  for(a=0;a<2;a++){
    if((cs[7][a]<2)&&(cs[8][a]<2)&&(cs[15][a]<2)&&(cs[16][a]<2)){
      nm[16]=a;
      cs[7][a]++; cs[8][a]++; cs[15][a]++; cs[16][a]++;
      stp17();
      cs[16][a]--; cs[15][a]--; cs[8][a]--; cs[7][a]--;
    }
  }
}
// Level #2:
// Set n17
void stp17(){
 short a;
  for(a=0;a<2;a++){
    if(bs[2][a]<4){
      if((cs[9][a]<2)&&(cs[16][a]<2)&&(cs[17][a]<2)&&(cs[24][a]<2)){
        nm[17]=a;
        bs[2][a]++;
        cs[9][a]++; cs[16][a]++; cs[17][a]++; cs[24][a]++;
        stp18();
        cs[24][a]--; cs[17][a]--; cs[16][a]--; cs[9][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n18
void stp18(){
 short a;
  for(a=0;a<2;a++){
    if((cs[9][a]<2)&&(cs[10][a]<2)&&(cs[17][a]<2)&&(cs[18][a]<2)){
      nm[18]=a;
      cs[9][a]++; cs[10][a]++; cs[17][a]++; cs[18][a]++;
```

```
      stp19();
      cs[18][a]--; cs[17][a]--; cs[10][a]--; cs[9][a]--;
    }
  }
}
// Set n19
void stp19(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[10][a]<2)&&(cs[11][a]<2)&&(cs[18][a]<2)&&(cs[19][a]<2)){
        nm[19]=a;
        bs[3][a]++;
        cs[10][a]++; cs[11][a]++; cs[18][a]++; cs[19][a]++;
        stp20();
        cs[19][a]--; cs[18][a]--; cs[11][a]--; cs[10][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n20
void stp20(){
 short a;
  for(a=0;a<2;a++){
    if((cs[11][a]<2)&&(cs[12][a]<2)&&(cs[19][a]<2)&&(cs[20][a]<2)){
      nm[20]=a;
      cs[11][a]++; cs[12][a]++; cs[19][a]++; cs[20][a]++;
      stp21();
      cs[20][a]--; cs[19][a]--; cs[12][a]--; cs[11][a]--;
    }
  }
}
// Set n21
void stp21(){
 short a;
  for(a=0;a<2;a++){
    if((cs[12][a]<2)&&(cs[13][a]<2)&&(cs[20][a]<2)&&(cs[21][a]<2)){
      nm[21]=a;
      cs[12][a]++; cs[13][a]++; cs[20][a]++; cs[21][a]++;
      stp22();
      cs[21][a]--; cs[20][a]--; cs[13][a]--; cs[12][a]--;
    }
  }
}
// Set n22
void stp22(){
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[13][a]<2)&&(cs[14][a]<2)&&(cs[21][a]<2)&&(cs[22][a]<2)){
        nm[22]=a;
        bs[4][a]++;
        cs[13][a]++; cs[14][a]++; cs[21][a]++; cs[22][a]++;
        stp23();
        cs[22][a]--; cs[21][a]--; cs[14][a]--; cs[13][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n23
void stp23(){
 short a;
  for(a=0;a<2;a++){
    if((cs[14][a]<2)&&(cs[15][a]<2)&&(cs[22][a]<2)&&(cs[23][a]<2)){
      nm[23]=a;
```

```
      cs[14][a]++; cs[15][a]++; cs[22][a]++; cs[23][a]++;
      stp24();
      cs[23][a]--; cs[22][a]--; cs[15][a]--; cs[14][a]--;
    }
  }
}
// Set n24
void stp24(){
 short a;
  for(a=0;a<2;a++){
    if((cs[15][a]<2)&&(cs[16][a]<2)&&(cs[23][a]<2)&&(cs[24][a]<2)){
      nm[24]=a;
      cs[15][a]++; cs[16][a]++; cs[23][a]++; cs[24][a]++;
      stp25();
      cs[24][a]--; cs[23][a]--; cs[16][a]--; cs[15][a]--;
    }
  }
}
// Level #3:
// Set n25
void stp25(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[2][a]<4){
      if((cs[17][a]<2)&&(cs[24][a]<2)&&(cs[25][a]<2)&&(cs[32][a]<2)){
        nm[25]=a;
        bs[2][a]++;
        cs[17][a]++; cs[24][a]++; cs[25][a]++; cs[32][a]++;
        stp26();
        cs[32][a]--; cs[25][a]--; cs[24][a]--; cs[17][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n26
void stp26(){
 short a;
  for(a=0;a<2;a++){
    if((cs[17][a]<2)&&(cs[18][a]<2)&&(cs[25][a]<2)&&(cs[26][a]<2)){
      nm[26]=a;
      cs[17][a]++; cs[18][a]++; cs[25][a]++; cs[26][a]++;
      stp27();
      cs[26][a]--; cs[25][a]--; cs[18][a]--; cs[17][a]--;
    }
  }
}
// Set n27
void stp27(){
 short a;
  for(a=0;a<2;a++){
    if((cs[18][a]<2)&&(cs[19][a]<2)&&(cs[26][a]<2)&&(cs[27][a]<2)){
      nm[27]=a;
      cs[18][a]++; cs[19][a]++; cs[26][a]++; cs[27][a]++;
      stp28();
      cs[27][a]--; cs[26][a]--; cs[19][a]--; cs[18][a]--;
    }
  }
}
// Set n28
void stp28(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[19][a]<2)&&(cs[20][a]<2)&&(cs[27][a]<2)&&(cs[28][a]<2)){
        nm[28]=a;
```

```
        bs[3][a]++;
        cs[19][a]++; cs[20][a]++; cs[27][a]++; cs[28][a]++;
        stp29();
        cs[28][a]--; cs[27][a]--; cs[20][a]--; cs[19][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n29
void stp29(){
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[20][a]<2)&&(cs[21][a]<2)&&(cs[28][a]<2)&&(cs[29][a]<2)){
        nm[29]=a;
        bs[4][a]++;
        cs[20][a]++; cs[21][a]++; cs[28][a]++; cs[29][a]++;
        stp30();
        cs[29][a]--; cs[28][a]--; cs[21][a]--; cs[20][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n30
void stp30(){
 short a;
  for(a=0;a<2;a++){
    if((cs[21][a]<2)&&(cs[22][a]<2)&&(cs[29][a]<2)&&(cs[30][a]<2)){
      nm[30]=a;
      cs[21][a]++; cs[22][a]++; cs[29][a]++; cs[30][a]++;
      stp31();
      cs[30][a]--; cs[29][a]--; cs[22][a]--; cs[21][a]--;
    }
  }
}
// Set n31
void stp31(){
 short a;
  for(a=0;a<2;a++){
    if((cs[22][a]<2)&&(cs[23][a]<2)&&(cs[30][a]<2)&&(cs[31][a]<2)){
      nm[31]=a;
      cs[22][a]++; cs[23][a]++; cs[30][a]++; cs[31][a]++;
      stp32();
      cs[31][a]--; cs[30][a]--; cs[23][a]--; cs[22][a]--;
    }
  }
}
// Set n32
void stp32(){
 short a;
  for(a=0;a<2;a++){
    if((cs[23][a]<2)&&(cs[24][a]<2)&&(cs[31][a]<2)&&(cs[32][a]<2)){
      nm[32]=a;
      cs[23][a]++; cs[24][a]++; cs[31][a]++; cs[32][a]++;
      stp33();
      cs[32][a]--; cs[31][a]--; cs[24][a]--; cs[23][a]--;
    }
  }
}
// Level #4:
// Set n33
void stp33(){
 short a;
  for(a=0;a<2;a++){
    if(bs[2][a]<4){
```

```
      if((cs[25][a]<2)&&(cs[32][a]<2)&&(cs[33][a]<2)&&(cs[40][a]<2)){
        nm[33]=a;
        bs[2][a]++;
        cs[25][a]++; cs[32][a]++; cs[33][a]++; cs[40][a]++;
        stp34();
        cs[40][a]--; cs[33][a]--; cs[32][a]--; cs[25][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n34
void stp34(){
 short a;
  for(a=0;a<2;a++){
    if((cs[25][a]<2)&&(cs[26][a]<2)&&(cs[33][a]<2)&&(cs[34][a]<2)){
      nm[34]=a;
      cs[25][a]++; cs[26][a]++; cs[33][a]++; cs[34][a]++;
      stp35();
      cs[34][a]--; cs[33][a]--; cs[26][a]--; cs[25][a]--;
    }
  }
}
// Set n35
void stp35(){
 short a;
  for(a=0;a<2;a++){
    if((cs[26][a]<2)&&(cs[27][a]<2)&&(cs[34][a]<2)&&(cs[35][a]<2)){
      nm[35]=a;
      cs[26][a]++; cs[27][a]++; cs[34][a]++; cs[35][a]++;
      stp36();
      cs[35][a]--; cs[34][a]--; cs[27][a]--; cs[26][a]--;
    }
  }
}
// Set n36
void stp36(){
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[27][a]<2)&&(cs[28][a]<2)&&(cs[35][a]<2)&&(cs[36][a]<2)){
        nm[36]=a;
        bs[4][a]++;
        cs[27][a]++; cs[28][a]++; cs[35][a]++; cs[36][a]++;
        stp37();
        cs[36][a]--; cs[35][a]--; cs[28][a]--; cs[27][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n37
void stp37(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[28][a]<2)&&(cs[29][a]<2)&&(cs[36][a]<2)&&(cs[37][a]<2)){
        nm[37]=a;
        bs[3][a]++;
        cs[28][a]++; cs[29][a]++; cs[36][a]++; cs[37][a]++;
        stp38();
        cs[37][a]--; cs[36][a]--; cs[29][a]--; cs[28][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n38
```

```
void stp38(){
 short a;
  for(a=0;a<2;a++){
    if((cs[29][a]<2)&&(cs[30][a]<2)&&(cs[37][a]<2)&&(cs[38][a]<2)){
      nm[38]=a;
      cs[29][a]++; cs[30][a]++; cs[37][a]++; cs[38][a]++;
      stp39();
      cs[38][a]--; cs[37][a]--; cs[30][a]--; cs[29][a]--;
    }
  }
}
// Set n39
void stp39(){
 short a;
  for(a=0;a<2;a++){
    if((cs[30][a]<2)&&(cs[31][a]<2)&&(cs[38][a]<2)&&(cs[39][a]<2)){
      nm[39]=a;
      cs[30][a]++; cs[31][a]++; cs[38][a]++; cs[39][a]++;
      stp40();
      cs[39][a]--; cs[38][a]--; cs[31][a]--; cs[30][a]--;
    }
  }
}
// Set n40
void stp40(){
 short a;
  for(a=0;a<2;a++){
    if((cs[31][a]<2)&&(cs[32][a]<2)&&(cs[39][a]<2)&&(cs[40][a]<2)){
      nm[40]=a;
      cs[31][a]++; cs[32][a]++; cs[39][a]++; cs[40][a]++;
      stp41();
      cs[40][a]--; cs[39][a]--; cs[32][a]--; cs[31][a]--;
    }
  }
}
// Level #5:
// Set n41
void stp41(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[2][a]<4){
      if((cs[33][a]<2)&&(cs[40][a]<2)&&(cs[41][a]<2)&&(cs[48][a]<2)){
        nm[41]=a;
        bs[2][a]++;
        cs[33][a]++; cs[40][a]++; cs[41][a]++; cs[48][a]++;
        stp42();
        cs[48][a]--; cs[41][a]--; cs[40][a]--; cs[33][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n42
void stp42(){
 short a;
  for(a=0;a<2;a++){
    if((cs[33][a]<2)&&(cs[34][a]<2)&&(cs[41][a]<2)&&(cs[42][a]<2)){
      nm[42]=a;
      cs[33][a]++; cs[34][a]++; cs[41][a]++; cs[42][a]++;
      stp43();
      cs[42][a]--; cs[41][a]--; cs[34][a]--; cs[33][a]--;
    }
  }
}
// Set n43
void stp43(){
```

```
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[34][a]<2)&&(cs[35][a]<2)&&(cs[42][a]<2)&&(cs[43][a]<2)){
        nm[43]=a;
        bs[4][a]++;
        cs[34][a]++; cs[35][a]++; cs[42][a]++; cs[43][a]++;
        stp44();
        cs[43][a]--; cs[42][a]--; cs[35][a]--; cs[34][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n44
void stp44(){
 short a;
  for(a=0;a<2;a++){
    if((cs[35][a]<2)&&(cs[36][a]<2)&&(cs[43][a]<2)&&(cs[44][a]<2)){
      nm[44]=a;
      cs[35][a]++; cs[36][a]++; cs[43][a]++; cs[44][a]++;
      stp45();
      cs[44][a]--; cs[43][a]--; cs[36][a]--; cs[35][a]--;
    }
  }
}
// Set n45
void stp45(){
 short a;
  for(a=0;a<2;a++){
    if((cs[36][a]<2)&&(cs[37][a]<2)&&(cs[44][a]<2)&&(cs[45][a]<2)){
      nm[45]=a;
      cs[36][a]++; cs[37][a]++; cs[44][a]++; cs[45][a]++;
      stp46();
      cs[45][a]--; cs[44][a]--; cs[37][a]--; cs[36][a]--;
    }
  }
}
// Set n46
void stp46(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[37][a]<2)&&(cs[38][a]<2)&&(cs[45][a]<2)&&(cs[46][a]<2)){
        nm[46]=a;
        bs[3][a]++;
        cs[37][a]++; cs[38][a]++; cs[45][a]++; cs[46][a]++;
        stp47();
        cs[46][a]--; cs[45][a]--; cs[38][a]--; cs[37][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n47
void stp47(){
 short a;
  for(a=0;a<2;a++){
    if((cs[38][a]<2)&&(cs[39][a]<2)&&(cs[46][a]<2)&&(cs[47][a]<2)){
      nm[47]=a;
      cs[38][a]++; cs[39][a]++; cs[46][a]++; cs[47][a]++;
      stp48();
      cs[47][a]--; cs[46][a]--; cs[39][a]--; cs[38][a]--;
    }
  }
}
// Set n48
```

```
void stp48(){
 short a;
  for(a=0;a<2;a++){
    if((cs[39][a]<2)&&(cs[40][a]<2)&&(cs[47][a]<2)&&(cs[48][a]<2)){
      nm[48]=a;
      cs[39][a]++; cs[40][a]++; cs[47][a]++; cs[48][a]++;
      stp49();
      cs[48][a]--; cs[47][a]--; cs[40][a]--; cs[39][a]--;
    }
  }
}
// Level #6:
// Set n49
void stp49(){
 short a;
  for(a=0;a<2;a++){
    if(bs[2][a]<4){
      if((cs[41][a]<2)&&(cs[48][a]<2)&&(cs[49][a]<2)&&(cs[56][a]<2)){
        nm[49]=a;
        bs[2][a]++;
        cs[41][a]++; cs[48][a]++; cs[49][a]++; cs[56][a]++;
        stp50();
        cs[56][a]--; cs[49][a]--; cs[48][a]--; cs[41][a]--;
        bs[2][a]--;
      }}
  }
}
// Set n57
void stp50(){
 short a;
  for(a=0;a<2;a++){
    if((bs[2][a]<4)&&(bs[4][a]<4)){
      if((cs[49][a]<2)&&(cs[56][a]<2)&&(cs[57][a]<2)&&(cs[64][a]<2)){
        nm[57]=a;
        bs[2][a]++; bs[4][a]++;
        cs[49][a]++; cs[56][a]++; cs[57][a]++; cs[64][a]++;
        stp51();
        cs[64][a]--; cs[57][a]--; cs[56][a]--; cs[49][a]--;
        bs[4][a]--; bs[2][a]--;
      }}
  }
}
// Set n50
void stp51(){
 short a;
  for(a=0;a<2;a++){
    if(bs[4][a]<4){
      if((cs[41][a]<2)&&(cs[42][a]<2)&&(cs[49][a]<2)&&(cs[50][a]<2)){
        nm[50]=a;
        bs[4][a]++;
        cs[41][a]++; cs[42][a]++; cs[49][a]++; cs[50][a]++;
        stp52();
        cs[50][a]--; cs[49][a]--; cs[42][a]--; cs[41][a]--;
        bs[4][a]--;
      }}
  }
}
// Set n51
void stp52(){
 short a;
  for(a=0;a<2;a++){
    if((cs[42][a]<2)&&(cs[43][a]<2)&&(cs[50][a]<2)&&(cs[51][a]<2)){
      nm[51]=a;
      cs[42][a]++; cs[43][a]++; cs[50][a]++; cs[51][a]++;
      stp53();
```

```
      cs[51][a]--; cs[50][a]--; cs[43][a]--; cs[42][a]--;
    }
  }
}
// Set n52
void stp53(){
 short a;
  for(a=0;a<2;a++){
    if((cs[43][a]<2)&&(cs[44][a]<2)&&(cs[51][a]<2)&&(cs[52][a]<2)){
      nm[52]=a;
      cs[43][a]++; cs[44][a]++; cs[51][a]++; cs[52][a]++;
      stp54();
      cs[52][a]--; cs[51][a]--; cs[44][a]--; cs[43][a]--;
    }
  }
}
// Set n53
void stp54(){
 short a;
  for(a=0;a<2;a++){
    if((cs[44][a]<2)&&(cs[45][a]<2)&&(cs[52][a]<2)&&(cs[53][a]<2)){
      nm[53]=a;
      cs[44][a]++; cs[45][a]++; cs[52][a]++; cs[53][a]++;
      stp55();
      cs[53][a]--; cs[52][a]--; cs[45][a]--; cs[44][a]--;
    }
  }
}
// Set n54
void stp55(){
 short a;
  for(a=0;a<2;a++){
    if((cs[45][a]<2)&&(cs[46][a]<2)&&(cs[53][a]<2)&&(cs[54][a]<2)){
      nm[54]=a;
      cs[45][a]++; cs[46][a]++; cs[53][a]++; cs[54][a]++;
      stp56();
      cs[54][a]--; cs[53][a]--; cs[46][a]--; cs[45][a]--;
    }
  }
}
// Set n55
void stp56(){
 short a;
  for(a=0;a<2;a++){
    if(bs[3][a]<4){
      if((cs[46][a]<2)&&(cs[47][a]<2)&&(cs[54][a]<2)&&(cs[55][a]<2)){
        nm[55]=a;
        bs[3][a]++;
        cs[46][a]++; cs[47][a]++; cs[54][a]++; cs[55][a]++;
        stp57();
        cs[55][a]--; cs[54][a]--; cs[47][a]--; cs[46][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n56
void stp57(){
 short a;
  for(a=0;a<2;a++){
    if((cs[47][a]<2)&&(cs[48][a]<2)&&(cs[55][a]<2)&&(cs[56][a]<2)){
      nm[56]=a;
      cs[47][a]++; cs[48][a]++; cs[55][a]++; cs[56][a]++;
      stp58();
      cs[56][a]--; cs[55][a]--; cs[48][a]--; cs[47][a]--;
    }
```

```
  }
}
// Level #7:
// Set n64
void stp58(){
 short a;
  for(a=1;a>=0;a--){
    if(bs[3][a]<4){
      if((cs[55][a]<2)&&(cs[56][a]<2)&&(cs[63][a]<2)&&(cs[64][a]<2)){
        nm[64]=a;
        bs[3][a]++;
        cs[55][a]++; cs[56][a]++; cs[63][a]++; cs[64][a]++;
        stp59();
        cs[64][a]--; cs[63][a]--; cs[56][a]--; cs[55][a]--;
        bs[3][a]--;
      }}
  }
}
// Set n58
void stp59(){
 short a;
  for(a=0;a<2;a++){
    if((cs[49][a]<2)&&(cs[50][a]<2)&&(cs[57][a]<2)&&(cs[58][a]<2)){
      nm[58]=a;
      cs[49][a]++; cs[50][a]++; cs[57][a]++; cs[58][a]++;
      stp60();
      cs[58][a]--; cs[57][a]--; cs[50][a]--; cs[49][a]--;
    }
  }
}
// Set n59
void stp60(){
 short a;
  for(a=0;a<2;a++){
    if((cs[50][a]<2)&&(cs[51][a]<2)&&(cs[58][a]<2)&&(cs[59][a]<2)){
      nm[59]=a;
      cs[50][a]++; cs[51][a]++; cs[58][a]++; cs[59][a]++;
      stp61();
      cs[59][a]--; cs[58][a]--; cs[51][a]--; cs[50][a]--;
    }
  }
}
// Set n60
void stp61(){
 short a;
  for(a=0;a<2;a++){
    if((cs[51][a]<2)&&(cs[52][a]<2)&&(cs[59][a]<2)&&(cs[60][a]<2)){
      nm[60]=a;
      cs[51][a]++; cs[52][a]++; cs[59][a]++; cs[60][a]++;
      stp62();
      cs[60][a]--; cs[59][a]--; cs[52][a]--; cs[51][a]--;
    }
  }
}
// Set n61
void stp62(){
 short a;
  for(a=0;a<2;a++){
    if((cs[52][a]<2)&&(cs[53][a]<2)&&(cs[60][a]<2)&&(cs[61][a]<2)){
      nm[61]=a;
      cs[52][a]++; cs[53][a]++; cs[60][a]++; cs[61][a]++;
      stp63();
      cs[61][a]--; cs[60][a]--; cs[53][a]--; cs[52][a]--;
    }
  }
```

```
}
// Set n62
void stp63(){
 short a;
  for(a=0;a<2;a++){
    if((cs[53][a]<2)&&(cs[54][a]<2)&&(cs[61][a]<2)&&(cs[62][a]<2)){
      nm[62]=a;
      cs[53][a]++; cs[54][a]++; cs[61][a]++; cs[62][a]++;
      stp64();
      cs[62][a]--; cs[61][a]--; cs[54][a]--; cs[53][a]--;
    }
  }
}
// Set n63
void stp64(){
 short a;
  for(a=0;a<2;a++){
    if((cs[54][a]<2)&&(cs[55][a]<2)&&(cs[62][a]<2)&&(cs[63][a]<2)){
      nm[63]=a;
      cs[54][a]++; cs[55][a]++; cs[62][a]++; cs[63][a]++;
      lurecord();
      cs[63][a]--; cs[62][a]--; cs[55][a]--; cs[54][a]--;
    }
  }
}
//
//* Record the Latin Units *
void lurecord(){
 short n;
  tlu[lcnt][0]=lcnt+1;
  for(n=1;n<65;n++){tlu[lcnt][n]=nm[n];}
  lcnt++;
}
//
//* Classify and Print the Latin Units *
void prlunit(){
 short t,m,n,l;
 short l8;
  for(m=0;m<lcnt;m++){
    for(n=0;n<lcnt;n++){
      t=0;
      for(l=1;l<65;l++){if(tlu[m][l]==tlu[n][l]){t++;}}
      mtc[m][n]=t;
    }
  }
  for(t=0;t<lcnt;t=t+9){
    printf("%9d/%9d/%9d/%9d/%9d/%9d/%9d/%9d/%9d/\n",
            t+1,t+2,t+3,t+4,t+5,t+6,t+7,t+8,t+9);
    for(l=0;l<8;l++){l8=l*8;
      for(m=t;m<(t+9);m++){
        printf("  ");
        for(n=1;n<9;n++){printf("%d",tlu[m][l8+n]);}
      }
      printf("\n");
    }
  }
  printf(" [Count of Latin Units = %d]\n",lcnt);
  printf("\n");
  printf(" [Reference Table of Matching Digits: 32 is the Best;]\n");
  printf(" #1|");
  for(n=1;n<=28;n++){printf("%3d",n);} printf("\n");
  printf(" --+");
  for(n=0;n<28;n++){printf("---");}; printf("\n");
  for(m=0;m<28;m++){
    printf("%3d|",m+1);
```

```
    for(n=0;n<28;n++){t=mtc[m][n];
      if(t>=0){printf("%3d",t);}else{printf("  -");}}
    printf("\n");
  }
  printf("\n");
  printf(" #2|");
  for(n=45;n<=lcnt;n++){printf("%3d",n);} printf("\n");
  printf(" --+");
  for(n=44;n<lcnt;n++){printf("---");}; printf("\n");
  for(m=44;m<lcnt;m++){
    printf("%3d|",m+1);
    for(n=44;n<lcnt;n++){t=mtc[m][n];
      if(t>=0){printf("%3d",t);}else{printf("  -");}}
    printf("\n");
  }
}
//
//** Combine & Compose **
void cmbcmp(){
  for(u1=0;u1<(lcnt/2);u1++){cnt2=0;
    cmcm2();}
}
//* Sub #2 for 'Combine & Compose' *
void cmcm2(){
  for(u2=0;u2<lcnt;u2++){
    if(mtc[u2][u1]==32){if(u1<1){cnt2=0;}
      cmcm3();}
  }
}
//* Sub #3 for 'Combine & Compose' *
void cmcm3(){
  for(u3=0;u3<lcnt;u3++){
    if(mtc[u3][u1]==32){
      if(mtc[u3][u2]==32){
        cmcm4();}
    }}
}
//* Sub #4 for 'Combine & Compose' *
void cmcm4(){
  for(u4=0;u4<lcnt;u4++){
    if(mtc[u4][u1]==32){
      if(mtc[u4][u2]==32){
        if(mtc[u4][u3]==32){cmcm5();}
      }}}
}
//* Sub #5 for 'Combine & Compose' *
void cmcm5(){
  for(u5=0;u5<lcnt;u5++){
    if(mtc[u5][u1]==32){
      if(mtc[u5][u2]==32){
        if(mtc[u5][u3]==32){
          if(mtc[u5][u4]==32){cmcm6();}
        }}}}
}
//* Sub #6 for 'Combine & Compose' *
void cmcm6(){
  for(u6=0;u6<lcnt;u6++){
    if(mtc[u6][u1]==32){
      if(mtc[u6][u2]==32){
        if(mtc[u6][u3]==32){
          if(mtc[u6][u4]==32){
            if(mtc[u6][u5]==32){cmcm7();}
          }}}}}
}
//* Sub #7 for 'Combine & Compose' *
```

```
void cmcm7(){
 short n,dt,fc;
  for(n=1;n<65;n++){uflg[n]=0;}
  for(n=1;n<65;n++){
    dt=tlu[u1][n]*32+tlu[u2][n]*16+tlu[u3][n]*8+tlu[u4][n]*4+tlu[u5][n]*2+tlu[u6][n]+1;
    nm[n]=dt; uflg[dt]++;}
  fc=0;
  for(n=1;n<65;n++){if(uflg[n]==1){fc++;}else{break;}}
  if(fc==64){
    if((nm[1]<nm[64])&&(nm[1]<nm[57])&&(nm[57]<nm[8])){solrecord();}
  }
}
//
//* Record the Answers only with n1=1 *
void solrecord(){
 short n;
  cnt++; n1c[nm[1]]++;
  if(nm[1]==1){
    cnt1++; cnt2++;
    if(cnt2==1){
      cntr[cnt3]=cnt1;
      for(n=1;n<65;n++){anm[cnt3][n]=nm[n];}
      anm[cnt3][65]=u1+1; anm[cnt3][66]=u2+1; anm[cnt3][67]=u3+1;
      anm[cnt3][68]=u4+1; anm[cnt3][69]=u5+1; anm[cnt3][70]=u6+1;
      cnt3++; if(cnt3==3){pr3sol(cnt3); cnt3=0;}}
  }
}
//
//* Print 3 Answers in 1 line *
void pr3sol(short x){
 short l,l8,m,n;
  for(m=0;m<x;m++){
    printf("%4d/%2d/%2d/%2d/%2d/%2d/%8ld#",
           anm[m][65],anm[m][66],anm[m][67],anm[m][68],anm[m][69],anm[m][70],cntr[m]);}
  printf("\n");
  for(l=0;l<8;l++){l8=l*8;
    for(m=0;m<x;m++){
      printf("    ");
      for(n=1;n<9;n++){printf("%3d",anm[m][l8+n]);}
      if(m+1<x){printf(" ");}
    }
    printf("\n");
  }
}
//
//* Print the Count according to n1 *
void prn1c(){
 short m;
  for(m=1;m<37;m++){
    printf("  [%2d]%8ld,",m,n1c[m]);
    if(m%6==0){printf("\n");}
  }
}
//
```

[List of **72 Euler Units** by the Binary Decompositions]

1/	2/	3/	4/	5/	6/	7/	8/	9/
01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010
01010101	01010101	01010101	10101010	10101010	10101010	10101010	10101010	10101010
10101010	01010101	01010101	10101010	10101010	01010101	01010101	01010101	01010101
10101010	10101010	10101010	01010101	10101010	01010101	01010101	10101010	10101010
01010101	10101010	01010101	01010101	01010101	10101010	01010101	10101010	01010101
10101010	10101010	10101010	10101010	01010101	10101010	10101010	01010101	01010101
01010101	01010101	10101010	01010101	01010101	01010101	10101010	01010101	10101010

10/	11/	12/	13/	14/	15/	16/	17/	18/
01011010	01001110	01001011	01110010	01111000	01100110	01100011	01101100	01101001
10100101	10110001	10110100	10001101	10000111	10011001	10011100	10010011	10010110
01011010	01001110	01001011	01110010	01111000	01100110	01100011	01101100	01101001
10100101	10110001	10110100	10001101	10000111	10011001	10011100	10010011	10010110
01011010	01001110	01001011	01110010	01111000	01100110	01100011	01101100	01101001
10100101	10110001	10110100	10001101	10000111	10011001	10011100	10010011	10010110
01011010	01001110	01001011	01110010	01111000	01100110	01100011	01101100	01101001
10100101	10110001	10110100	10001101	10000111	10011001	10011100	10010011	10010110

19/	20/	21/	22/	23/	24/	25/	26/	27/
01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
01010101	01010101	01010101	10101010	10101010	10101010	10101010	10101010	10101010
10101010	10101010	01010101	10101010	10101010	10101010	10101010	01010101	01010101
10101010	10101010	10101010	01010101	01010101	10101010	10101010	01010101	10101010
10101010	01010101	10101010	10101010	01010101	10101010	01010101	10101010	10101010
10101010	10101010	10101010	10101010	10101010	01010101	01010101	10101010	01010101
01010101	10101010	10101010	01010101	10101010	01010101	10101010	10101010	10101010

28/	29/	30/	31/	32/	33/	34/	35/	36/
00011110	00011011	00001111	00110110	00110011	00111100	00111001	00100111	00101101
11100001	11100100	11110000	11001001	11001100	11000011	11000110	11011000	11010010
00011110	00011011	00001111	00110110	00110011	00111100	00111001	00100111	00101101
11100001	11100100	11110000	11001001	11001100	11000011	11000110	11011000	11010010
00011110	00011011	00001111	00110110	00110011	00111100	00111001	00100111	00101101
11100001	11100100	11110000	11001001	11001100	11000011	11000110	11011000	11010010
00011110	00011011	00001111	00110110	00110011	00111100	00111001	00100111	00101101
11100001	11100100	11110000	11001001	11001100	11000011	11000110	11011000	11010010

37/	38/	39/	40/	41/	42/	43/	44/	45/
11010010	11011000	11000110	11000011	11001100	11001001	11110000	11100100	11100001
00101101	00100111	00111001	00111100	00110011	00110110	00001111	00011011	00011110
11010010	11011000	11000110	11000011	11001100	11001001	11110000	11100100	11100001
00101101	00100111	00111001	00111100	00110011	00110110	00001111	00011011	00011110
11010010	11011000	11000110	11000011	11001100	11001001	11110000	11100100	11100001
00101101	00100111	00111001	00111100	00110011	00110110	00001111	00011011	00011110
11010010	11011000	11000110	11000011	11001100	11001001	11110000	11100100	11100001
00101101	00100111	00111001	00111100	00110011	00110110	00001111	00011011	00011110

46/	47/	48/	49/	50/	51/	52/	53/	54/
10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010
10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010
01010101	01010101	01010101	01010101	01010101	01010101	10101010	10101010	10101010
10101010	10101010	01010101	01010101	01010101	01010101	10101010	01010101	01010101
01010101	10101010	01010101	01010101	10101010	10101010	01010101	01010101	01010101
01010101	01010101	10101010	01010101	10101010	01010101	01010101	10101010	01010101
10101010	01010101	10101010	10101010	01010101	01010101	01010101	01010101	01010101
01010101	01010101	01010101	10101010	01010101	10101010	01010101	01010101	10101010

55/	56/	57/	58/	59/	60/	61/	62/	63/
10010110	10010011	10011100	10011001	10000111	10001101	10110100	10110001	10100101
01101001	01101100	01100011	01100110	01111000	01110010	01001011	01001110	01011010
10010110	10010011	10011100	10011001	10000111	10001101	10110100	10110001	10100101
01101001	01101100	01100011	01100110	01111000	01110010	01001011	01001110	01011010
10010110	10010011	10011100	10011001	10000111	10001101	10110100	10110001	10100101
01101001	01101100	01100011	01100110	01111000	01110010	01001011	01001110	01011010
10010110	10010011	10011100	10011001	10000111	10001101	10110100	10110001	10100101
01101001	01101100	01100011	01100110	01111000	01110010	01001011	01001110	01011010

64/	65/	66/	67/	68/	69/	70/	71/	72/
10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010	10101010
01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101	01010101
01010101	01010101	01010101	01010101	01010101	01010101	10101010	10101010	10101010
10101010	10101010	10101010	10101010	01010101	01010101	10101010	10101010	01010101
01010101	01010101	10101010	10101010	01010101	10101010	01010101	01010101	01010101
10101010	01010101	10101010	01010101	10101010	10101010	10101010	01010101	10101010
10101010	10101010	01010101	01010101	10101010	01010101	01010101	01010101	01010101
01010101	10101010	01010101	10101010	10101010	10101010	01010101	10101010	10101010

The next list above shows all of 72 Binary Units I have got.
Woo! Do we have to examine all cases combined and composed as many as about 36*35*34*33*32*31*26 (=89754255360) ?

6-2. Combine 6 Latin Squares to Compose each Solution

(1) Make the Reference Table on mtc[73][73], where every data is recorded about how many digits are the same between any two units picked up.

(2) Combine 6 units chosen and take them for the 6 layers of Binary Decompositions. Consult any unit pair with the Reference Table in advance to know the value of that pair is equal to 32. If it is true, you may calculate and compose each solution.

(3) Examine if your answer is against the **Def (3)** of C.E.S. or not. When it is all right, select under the List-forming Conditions, and at last you can get what you want.

```
[Compositions of 'Composite & Pan-diagonal' Magic Squares 8x8:
  Solutions(n1=1) with Used_Units////// Solutions No#; Sum_Checks||\/|]
     1/      6/      9/     10/     15/     21/                    1#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 01010101  1 64  3 62  5 60  7 58|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 01010101 63  2 61  4 59  6 57  8|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 01010101 25 40 27 38 29 36 31 34|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 01010101 39 26 37 28 35 30 33 32|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 10101010 42 23 44 21 46 19 48 17|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 10101010 24 41 22 43 20 45 18 47|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 10101010 50 15 52 13 54 11 56  9|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 10101010 16 49 14 51 12 53 10 55|260|260\260/260|
     1/      6/      9/     10/     15/     23/                    2#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 01010101  1 64  3 62  5 60  7 58|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 01010101 63  2 61  4 59  6 57  8|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 10101010 26 39 28 37 30 35 32 33|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 10101010 40 25 38 27 36 29 34 31|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 01010101 41 24 43 22 45 20 47 18|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 01010101 23 42 21 44 19 46 17 48|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 10101010 50 15 52 13 54 11 56  9|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 10101010 16 49 14 51 12 53 10 55|260|260\260/260|
     1/      6/      9/     10/     15/     24/                    3#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 01010101  1 64  3 62  5 60  7 58|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 01010101 63  2 61  4 59  6 57  8|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 10101010 26 39 28 37 30 35 32 33|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 10101010 40 25 38 27 36 29 34 31|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 10101010 42 23 44 21 46 19 48 17|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 10101010 24 41 22 43 20 45 18 47|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 01010101 49 16 51 14 53 12 55 10|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 01010101 15 50 13 52 11 54  9 56|260|260\260/260|
     1/      6/      9/     10/     15/     30/                    4#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 00001111  1 63  3 61  6 60  8 58|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 11110000 64  2 62  4 59  5 57  7|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 00001111 25 39 27 37 30 36 32 34|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 11110000 40 26 38 28 35 29 33 31|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 00001111 41 23 43 21 46 20 48 18|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 11110000 24 42 22 44 19 45 17 47|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 00001111 49 15 51 13 54 12 56 10|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 11110000 16 50 14 52 11 53  9 55|260|260\260/260|
     1/      6/      9/     10/     15/     32/                    5#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 00110011  1 63  4 62  5 59  8 58|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 11001100 64  2 61  3 60  6 57  7|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 00110011 25 39 28 38 29 35 32 34|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 11001100 40 26 37 27 36 30 33 31|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 00110011 41 23 44 22 45 19 48 18|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 11001100 24 42 21 43 20 46 17 47|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 00110011 49 15 52 14 53 11 56 10|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 11001100 16 50 13 51 12 54  9 55|260|260\260/260|
```

```
      1/       6/       9/      10/      15/      33/                        6#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100110 00111100  1 63  4 62  6 60  7 57|260|260\260/260|
10101010 10101010 10101010 10100101 10011001 11000011 64  2 61  3 59  5 58  8|260|260\260/260|
01010101 10101010 10101010 01011010 01100110 00111100 25 39 28 38 30 36 31 33|260|260\260/260|
10101010 01010101 01010101 10100101 10011001 11000011 40 26 37 27 35 29 34 32|260|260\260/260|
10101010 01010101 10101010 01011010 01100110 00111100 41 23 44 22 46 20 47 17|260|260\260/260|
01010101 10101010 01010101 10100101 10011001 11000011 24 42 21 43 19 45 18 48|260|260\260/260|
10101010 10101010 01010101 01011010 01100110 00111100 49 15 52 14 54 12 55  9|260|260\260/260|
01010101 01010101 10101010 10100101 10011001 11000011 16 50 13 51 11 53 10 56|260|260\260/260|

      1/       6/       9/      10/      16/      21/                        7#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 01010101  1 64  3 62  5 58  7 60|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 01010101 63  2 61  4 59  8 57  6|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 01010101 25 40 27 38 29 34 31 36|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 01010101 39 26 37 28 35 32 33 30|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 10101010 42 23 44 21 46 17 48 19|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 10101010 24 41 22 43 20 47 18 45|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 10101010 50 15 52 13 54  9 56 11|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 10101010 16 49 14 51 12 55 10 53|260|260\260/260|

      1/       6/       9/      10/      16/      23/                        8#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 01010101  1 64  3 62  5 58  7 60|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 01010101 63  2 61  4 59  8 57  6|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 10101010 26 39 28 37 30 33 32 35|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 10101010 40 25 38 27 36 31 34 29|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 01010101 41 24 43 22 45 18 47 20|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 01010101 23 42 21 44 19 48 17 46|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 10101010 50 15 52 13 54  9 56 11|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 10101010 16 49 14 51 12 55 10 53|260|260\260/260|

      1/       6/       9/      10/      16/      24/                        9#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 01010101  1 64  3 62  5 58  7 60|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 01010101 63  2 61  4 59  8 57  6|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 10101010 26 39 28 37 30 33 32 35|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 10101010 40 25 38 27 36 31 34 29|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 10101010 42 23 44 21 46 17 48 19|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 10101010 24 41 22 43 20 47 18 45|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 01010101 49 16 51 14 53 10 55 12|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 01010101 15 50 13 52 11 56  9 54|260|260\260/260|

      1/       6/       9/      10/      16/      30/                       10#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 00001111  1 63  3 61  6 58  8 60|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 11110000 64  2 62  4 59  7 57  5|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 00001111 25 39 27 37 30 34 32 36|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 11110000 40 26 38 28 35 31 33 29|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 00001111 41 23 43 21 46 18 48 20|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 11110000 24 42 22 44 19 47 17 45|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 00001111 49 15 51 13 54 10 56 12|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 11110000 16 50 14 52 11 55  9 53|260|260\260/260|

      1/       6/       9/      10/      16/      31/                       11#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 00110110  1 63  4 62  5 58  8 59|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 11001001 64  2 61  3 60  7 57  6|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 00110110 25 39 28 38 29 34 32 35|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 11001001 40 26 37 27 36 31 33 30|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 00110110 41 23 44 22 45 18 48 19|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 11001001 24 42 21 43 20 47 17 46|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 00110110 49 15 52 14 53 10 56 11|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 11001001 16 50 13 51 12 55  9 54|260|260\260/260|

      1/       6/       9/      10/      16/      34/                       12#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01100011 00111001  1 63  4 62  6 57  7 60|260|260\260/260|
10101010 10101010 10101010 10100101 10011100 11000110 64  2 61  3 59  8 58  5|260|260\260/260|
01010101 10101010 10101010 01011010 01100011 00111001 25 39 28 38 30 33 31 36|260|260\260/260|
10101010 01010101 01010101 10100101 10011100 11000110 40 26 37 27 35 32 34 29|260|260\260/260|
10101010 01010101 10101010 01011010 01100011 00111001 41 23 44 22 46 17 47 20|260|260\260/260|
01010101 10101010 01010101 10100101 10011100 11000110 24 42 21 43 19 48 18 45|260|260\260/260|
10101010 10101010 01010101 01011010 01100011 00111001 49 15 52 14 54  9 55 12|260|260\260/260|
01010101 01010101 10101010 10100101 10011100 11000110 16 50 13 51 11 56 10 53|260|260\260/260|
```

```
     1/       6/       9/      10/      17/      21/                    13#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01101100 01010101  1 64  3 62  7 60  5 58|260|260\260/260|
10101010 10101010 10101010 10100101 10010011 01010101 63  2 61  4 57  6 59  8|260|260\260/260|
01010101 10101010 10101010 01011010 01101100 01010101 25 40 27 38 31 36 29 34|260|260\260/260|
10101010 01010101 01010101 10100101 10010011 01010101 39 26 37 28 33 30 35 32|260|260\260/260|
10101010 01010101 10101010 01011010 01101100 10101010 42 23 44 21 48 19 46 17|260|260\260/260|
01010101 10101010 01010101 10100101 10010011 10101010 24 41 22 43 18 45 20 47|260|260\260/260|
10101010 10101010 01010101 01011010 01101100 10101010 50 15 52 13 56 11 54  9|260|260\260/260|
01010101 01010101 10101010 10100101 10010011 10101010 16 49 14 51 10 53 12 55|260|260\260/260|

     1/       6/       9/      10/      17/      23/                    14#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01101100 01010101  1 64  3 62  7 60  5 58|260|260\260/260|
10101010 10101010 10101010 10100101 10010011 01010101 63  2 61  4 57  6 59  8|260|260\260/260|
01010101 10101010 10101010 01011010 01101100 10101010 26 39 28 37 32 35 30 33|260|260\260/260|
10101010 01010101 01010101 10100101 10010011 10101010 40 25 38 27 34 29 36 31|260|260\260/260|
10101010 01010101 10101010 01011010 01101100 01010101 41 24 43 22 47 20 45 18|260|260\260/260|
01010101 10101010 01010101 10100101 10010011 01010101 23 42 21 44 17 46 19 48|260|260\260/260|
10101010 10101010 01010101 01011010 01101100 10101010 50 15 52 13 56 11 54  9|260|260\260/260|
01010101 01010101 10101010 10100101 10010011 10101010 16 49 14 51 10 53 12 55|260|260\260/260|

     1/       6/       9/      10/      17/      24/                    15#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01101100 01010101  1 64  3 62  7 60  5 58|260|260\260/260|
10101010 10101010 10101010 10100101 10010011 01010101 63  2 61  4 57  6 59  8|260|260\260/260|
01010101 10101010 10101010 01011010 01101100 10101010 26 39 28 37 32 35 30 33|260|260\260/260|
10101010 01010101 01010101 10100101 10010011 10101010 40 25 38 27 34 29 36 31|260|260\260/260|
10101010 01010101 10101010 01011010 01101100 10101010 42 23 44 21 48 19 46 17|260|260\260/260|
01010101 10101010 01010101 10100101 10010011 10101010 24 41 22 43 18 45 20 47|260|260\260/260|
10101010 10101010 01010101 01011010 01101100 01010101 49 16 51 14 55 12 53 10|260|260\260/260|
01010101 01010101 10101010 10100101 10010011 01010101 15 50 13 52  9 54 11 56|260|260\260/260|

     1/       6/       9/      10/      17/      30/                    16#|Row|Clm\Pd1/Pd2|
01010101 01010101 01010101 01011010 01101100 00001111  1 63  3 61  8 60  6 58|260|260\260/260|
10101010 10101010 10101010 10100101 10010011 11110000 64  2 62  4 57  5 59  7|260|260\260/260|
01010101 10101010 10101010 01011010 01101100 00001111 25 39 27 37 32 36 30 34|260|260\260/260|
10101010 01010101 01010101 10100101 10010011 11110000 40 26 38 28 33 29 35 31|260|260\260/260|
10101010 01010101 10101010 01011010 01101100 00001111 41 23 43 21 48 20 46 18|260|260\260/260|
01010101 10101010 01010101 10100101 10010011 11110000 24 42 22 44 17 45 19 47|260|260\260/260|
10101010 10101010 01010101 01011010 01101100 00001111 49 15 51 13 56 12 54 10|260|260\260/260|
01010101 01010101 10101010 10100101 10010011 11110000 16 50 14 52  9 53 11 55|260|260\260/260|
```

```
 1/ 7/ 8/10/15/21/   18433#    1/ 8/ 7/10/15/21/   36865#    1/ 9/ 6/10/15/21/   55297#
   1 64  3 62  5 60  7 58        1 64  3 62  5 60  7 58        1 64  3 62  5 60  7 58
  63  2 61  4 59  6 57  8       63  2 61  4 59  6 57  8       63  2 61  4 59  6 57  8
  25 40 27 38 29 36 31 34       25 40 27 38 29 36 31 34       25 40 27 38 29 36 31 34
  39 26 37 28 35 30 33 32       39 26 37 28 35 30 33 32       39 26 37 28 35 30 33 32
  42 23 44 21 46 19 48 17       50 15 52 13 54 11 56  9       50 15 52 13 54 11 56  9
  16 49 14 51 12 53 10 55       24 41 22 43 20 45 18 47       16 49 14 51 12 53 10 55
  50 15 52 13 54 11 56  9       42 23 44 21 46 19 48 17       42 23 44 21 46 19 48 17
  24 41 22 43 20 45 18 47       16 49 14 51 12 53 10 55       24 41 22 43 20 45 18 47

 1/10/ 6/ 9/15/21/   73729#    1/11/ 6/ 9/13/21/   86785#    1/12/ 6/ 9/13/21/   99841#
   1 64  3 62 17 48 19 46        1 64  3 48 17 62 19 46        1 64  3 48 17 46 19 62
  63  2 61  4 47 18 45 20       63  2 61 18 47  4 45 20       63  2 61 18 47 20 45  4
  13 52 15 50 29 36 31 34       13 52 15 36 29 50 31 34       13 52 15 36 29 34 31 50
  51 14 49 16 35 30 33 32       51 14 49 30 35 16 33 32       51 14 49 30 35 32 33 16
  38 27 40 25 54 11 56  9       38 27 40 11 54 25 56  9       38 27 40 11 54  9 56 25
  28 37 26 39 12 53 10 55       28 37 26 53 12 39 10 55       28 37 26 53 12 55 10 39
  42 23 44 21 58  7 60  5       42 23 44  7 58 21 60  5       42 23 44  7 58  5 60 21
  24 41 22 43  8 57  6 59       24 41 22 57  8 43  6 59       24 41 22 57  8 59  6 43

 1/13/ 6/ 9/11/21/  112897#    1/14/ 6/ 9/11/21/  125953#    1/15/ 6/ 9/10/21/  139009#
   1 64 17 62  3 48 19 46        1 64 17 62 19 48  3 46        1 64 17 48  3 62 19 46
  63  2 47  4 61 18 45 20       63  2 47  4 45 18 61 20       63  2 47 18 61  4 45 20
  13 52 29 50 15 36 31 34       13 52 29 50 31 36 15 34       13 52 29 36 15 50 31 34
  51 14 35 16 49 30 33 32       51 14 35 16 33 30 49 32       51 14 35 30 49 16 33 32
  38 27 54 25 40 11 56  9       38 27 54 25 56 11 40  9       38 27 54 11 40 25 56  9
  28 37 12 39 26 53 10 55       28 37 12 39 10 53 26 55       28 37 12 53 26 39 10 55
  42 23 58 21 44  7 60  5       42 23 58 21 60  7 44  5       42 23 58  7 44 21 60  5
  24 41  8 43 22 57  6 59       24 41  8 43  6 57 22 59       24 41  8 57 22 43  6 59
```

1/16/ 6/ 9/10/21/ 152065#

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

1/17/ 6/ 9/10/21/ 165121#

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

1/18/ 6/ 9/10/21/ 178177#

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

1/21/ 6/ 9/10/15/ 191233#

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

1/22/ 7/ 8/10/15/ 223489#

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

1/23/ 6/ 9/10/15/ 241921#

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

1/24/ 6/ 9/10/15/ 260353#

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

1/25/ 7/ 8/10/15/ 278785#

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

1/28/ 6/ 9/12/13/ 297217#

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

1/29/ 6/ 9/11/13/ 310273#

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

1/30/ 6/ 9/10/15/ 323329#

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

1/31/ 6/ 9/10/16/ 336385#

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

1/32/ 6/ 9/10/15/ 349441#

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

1/33/ 6/ 9/10/15/ 362497#

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

1/34/ 6/ 9/10/16/ 375553#

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

1/35/ 6/ 9/11/13/ 388609#

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

1/36/ 6/ 9/11/14/ 401665#

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

2/ 4/ 9/10/15/20/ 414721#

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

3/ 4/ 8/10/15/19/ 829441#

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

4/ 2/ 9/10/15/20/ 1026289#

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

5/ 2/ 7/10/15/20/ 1441009#

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

6/ 1/ 9/10/15/21/ 1855729#

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

7/ 1/ 8/10/15/21/ 2270449#

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

8/ 1/ 7/10/15/21/ 2467297#

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

9/ 1/ 6/10/15/21/ 2882017#

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

12/ 1/ 6/ 9/13/21/ 3078865#

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

16/ 1/ 6/ 9/10/21/ 3296737#

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

18/ 1/ 6/ 9/10/21/ 3514609#

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

19/ 3/ 4/ 8/10/15/ 3732481#

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

20/ 2/ 4/ 9/10/15/ 4147201#

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

21/ 1/ 6/ 9/10/15/ 4344049#

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

22/ 1/ 7/ 8/10/15/ 4540897#

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

23/ 1/ 6/ 9/10/15/ 4955617#

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

24/ 1/ 6/ 9/10/15/ 5152465#

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

25/ 1/ 7/ 8/10/15/ 5567185#

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

26/ 2/ 4/ 9/10/15/ 5764033#

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

27/ 2/ 5/ 7/10/15/ 5960881#

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

29/ 1/ 6/ 9/11/13/ 6157729#

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

30/ 1/ 6/ 9/10/15/ 6375601#

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

32/ 1/ 6/ 9/10/15/ 6593473#

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

34/ 1/ 6/ 9/10/16/ 6811345#

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

35/ 1/ 6/ 9/11/13/ 7029217#

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

```
 36/ 1/ 6/ 9/11/14/ 7247089#
    1 32 34 30 36 63  3 61
   64 33 31 35 29  2 62  4
   13 20 46 18 48 51 15 49
   52 45 19 47 17 14 50 16
   21 12 54 10 56 43 23 41
   44 53 11 55  9 22 42 24
   25  8 58  6 60 39 27 37      . . . . .
   40 57  7 59  5 26 38 28     [Solution Counts(n1=1)/Total = 7464960/119439360]

 [List of Counts according to the Value of n1]
 [ 1] 7464960,  [ 2] 7387200,  [ 3] 7275744,  [ 4] 7197984,  [ 5] 6954336,  [ 6] 6876576,
 [ 7] 6765120,  [ 8] 6687360,  [ 9] 5857920,  [10] 5780160,  [11] 5668704,  [12] 5590944,
 [13] 5347296,  [14] 5269536,  [15] 5158080,  [16] 5080320,  [17] 2384640,  [18] 2306880,
 [19] 2195424,  [20] 2117664,  [21] 1874016,  [22] 1796256,  [23] 1684800,  [24] 1607040,
 [25]  777600,  [26]  699840,  [27]  588384,  [28]  510624,  [29]  266976,  [30]  189216,
 [31]   77760,  [32]       0,  [33]       0,  [34]       0,  [35]       0,  [36]       0,
 . . . . .   [OK!]
** Calculated and Listed by Kanji Setsuda
    on May 13, 2012 with MacOSX 10.7.4 & Xcode 4.3.2 **
```

I have got it at last! Though it took about two hours for my Macbook to count them up through, this program ran very fast, faster than any other old methods I have ever had. I was really surprised. For I had never imagined I could know the total counts.

I could also calculate Fundamental 360 Solutions of 'C&P' MS88 under the same conditions with the ones of 'C&C' MS88. Each result was the same with the one I got before.

Everything tells us that any solution of 'C&P' MS88 proves to be always 'Complete Euler Square' of order 8, and the solution set of 'C&P' MS88 makes the subset of the one of 'C.E.S.' of order 8.

The factors are supposed to divide the next total counts of solutions as follows:

Type/	'C&C'MS88	Factors	'C&P'MS88
Fundamental	90	90*4=360	360
Total	368640	*4*81 =	119439360
Factors	90*64*64	*4*81 =	360*64*64*81

7. [To be Continued]

Originally written on August 23, 2003, April 23, 2006;
Revised on June 14, 2012; Working with MacOSX 10.7.4 & Xcode 4.3.2

By Kanji Setsuda: E-Mail Address: jag12001@nifty.com
http://homepage2.nifty.com/KanjiSetsuda/pages/EnglishP1.html