Part 3. New Algebraic Studies of Magic Squares: **Kanji Setsuda**

Chapter 1. New Algebraic Study of Magic Squares of Order 3

Section 1. The Standard Magic Type of Order 3

1. You may probably know about the general concept of magic squares of order 3.
We want any object which has such properties as follows, using each of the serial natural numbers {1, 2, 3, 4, 5, 6, 7, 8 and 9} strictly once for the same solution.

```
* Basic Form and Simultaneous Equations *
  .--.--.--.   n1+n2+n3=C ... eq1; ('C' means the Magic Constant.)
  |n1|n2|n3|   n4+n5+n6=C ... eq2;
  |--+--+--|   n7+n8+n9=C ... eq3;
  |n4|n5|n6|   n1+n4+n7=C ... eq4;
  |--+--+--|   n2+n5+n8=C ... eq5;
  |n7|n8|n9|   n3+n6+n9=C ... eq6;
  '--'--'--'   n1+n5+n9=C ... eq7; (Both eq7 and eq8 denote
               n3+n5+n7=C ... eq8;     the primary diagonals.)
```

We can determine the real value of magic constant C before all by some simple algebraic calculations as follows.

```
eq1+eq2+eq3
 n1+n2+n3+n4+n5+n6+n7+n8+n9 =3*C
  1+ 2+ 3+ 4+ 5+ 6+ 7+ 8+ 9 =3*C
                  (1+9)*9/2 =3*C
  Therefore                C = 15 ... r0
```

2. What does an actual solution look like? How many solutions in all can we find?
Let's make our personal computer calculate under those definitions above and print out all the solutions, shall we? I will show you a sample program for that.
I dictated it with the program language 'C' in such a simple, primitive style that all of you, beginners, could understand its content easily.

```
/* Primitive PPROGRAM #1 for Magic Squares of Order 3 */
/** 'MS33Prmtv.c' built by Kanji Setsuda **/
/**  Jan.20, 2005; on MacOSX & Xcode 1.5 **/
/**/
#include <stdio.h>
/**/
short cnt, LSM;
short n1,n2,n3,n4,n5,n6,n7,n8,n9;
short uflg[10];
/**/
/* Main Program */
int main(){
 short n;
  printf("\n*** Magic Squares of Order 3 ***\n\n");
  for(n=0;n<10;n++){uflg[n]=0;}
  LSM=15; cnt=0;
  for(n1=1;n1<10;n1++){
   if(uflg[n1]==0){uflg[n1]=1;
    for(n2=1;n2<10;n2++){
     if(uflg[n2]==0){uflg[n2]=1;
      for(n3=1;n3<10;n3++){
       if((n1+n2+n3==LSM)&&(uflg[n3]==0)){uflg[n3]=1;
```

```
        for(n4=1;n4<10;n4++){
         if(uflg[n4]==0){uflg[n4]=1;
          for(n7=1;n7<10;n7++){
           if((n1+n4+n7==LSM)&&(uflg[n7]==0)){uflg[n7]=1;
            for(n5=1;n5<10;n5++){
             if((n3+n5+n7==LSM)&&(uflg[n5]==0)){uflg[n5]=1;
              for(n6=1;n6<10;n6++){
               if((n4+n5+n6==LSM)&&(uflg[n6]==0)){uflg[n6]=1;
                for(n8=1;n8<10;n8++){
                 if((n2+n5+n8==LSM)&&(uflg[n8]==0)){uflg[n8]=1;
                  for(n9=1;n9<10;n9++){
                   if((n1+n5+n9==LSM)&&(n3+n6+n9==LSM)&&(n7+n8+n9==LSM)){
                    uflg[n9]=1;
                    cnt++;
                    printf("[%d]\n",cnt);
                    printf("%3d%3d%3d\n",n1,n2,n3);
                    printf("%3d%3d%3d\n",n4,n5,n6);
                    printf("%3d%3d%3d\n",n7,n8,n9);
                    printf("\n");
                    uflg[n9]=0;}}
                  uflg[n8]=0;}}
                uflg[n6]=0;}}
              uflg[n5]=0;}}
            uflg[n7]=0;}}
          uflg[n4]=0;}}
        uflg[n3]=0;}}
      uflg[n2]=0;}}
    uflg[n1]=0;}}
  printf("* Count = %d\n",cnt);
  printf(" OK!\n");
 return 0;
}
/**/
```

This program should output the next list of 'primitive' solutions for our object.

```
*** Magic Squares of Order 3 ***

[1]
  2  7  6
  9  5  1
  4  3  8

[2]
  2  9  4
  7  5  3
  6  1  8

[3]
  4  3  8
  9  5  1
  2  7  6

[4]
  4  9  2
  3  5  7
  8  1  6

[5]
  6  1  8
  7  5  3
  2  9  4

[6]
  6  7  2
  1  5  9
  8  3  4

[7]
  8  1  6
  3  5  7
  4  9  2

[8]
  8  3  4
  1  5  9
  6  7  2

* Count = 8
 OK!
```

They look different from each other. I would like to call those eight as 'primitive solutions', since we made them without any List-forming Inequality Conditions.

3. What does the result really mean? What properties do they actually have?

Let's examine each piece and know what it exactly is.

```
** Examine Each Primitive Solution: **
[1]
   .---.---.---. Rows & Columns  Pan-diagonals
 6 | 2 | 7 | 6 | 2   2+7+6=15;  2+5+8=15;
   |---+---+---|     9+5+1=15;  7+1+4=12;
 1 | 9 | 5 | 1 | 9   4+3+8=15;  6+9+3=18;
   |---+---+---|     2+9+4=15;  2+1+3= 6;
 8 | 4 | 3 | 8 | 4   7+5+3=15;  7+9+8=24;
   '---'---'---'     6+1+8=15;  6+5+4=15;
* Complementary Pairs:
    2+8=10; 7+3=10; 6+4=10; 9+1=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[2]
   .---.---.---. Rows & Columns  Pan-diagonals
 4 | 2 | 9 | 4 | 2   2+9+4=15;  2+5+8=15;
   |---+---+---|     7+5+3=15;  9+3+6=18;
 3 | 7 | 5 | 3 | 7   6+1+8=15;  4+7+1=12;
   |---+---+---|     2+7+6=15;  2+3+1= 6;
 8 | 6 | 1 | 8 | 6   9+5+1=15;  9+7+8=24;
   '---'---'---'     4+3+8=15;  4+5+6=15;
* Complementary Pairs:
    2+8=10; 9+1=10; 4+6=10; 7+3=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[3]
   .---.---.---. Rows & Columns  Pan-diagonals
 8 | 4 | 3 | 8 | 4   4+3+8=15;  4+5+6=15;
   |---+---+---|     9+5+1=15;  3+1+2= 6;
 1 | 9 | 5 | 1 | 9   2+7+6=15;  8+9+7=24;
   |---+---+---|     4+9+2=15;  4+1+7=12;
 6 | 2 | 7 | 6 | 2   3+5+7=15;  3+9+6=18;
   '---'---'---'     8+1+6=15;  8+5+2=15;
* Complementary Pairs:
    4+6=10; 3+7=10; 8+2=10; 9+1=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[4]
   .---.---.---. Rows & Columns  Pan-diagonals
 2 | 4 | 9 | 2 | 4   4+9+2=15;  4+5+6=15;
   |---+---+---|     3+5+7=15;  9+7+8=24;
 7 | 3 | 5 | 7 | 3   8+1+6=15;  2+3+1= 6;
   |---+---+---|     4+3+8=15;  4+7+1=12;
 6 | 8 | 1 | 6 | 8   9+5+1=15;  9+3+6=18;
   '---'---'---'     2+7+6=15;  2+5+8=15;
* Complementary Pairs:
    4+6=10; 9+1=10; 2+8=10; 3+7=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[5]
   .---.---.---. Rows & Columns  Pan-diagonals
 8 | 6 | 1 | 8 | 6   6+1+8=15;  6+5+4=15;
   |---+---+---|     7+5+3=15;  1+3+2= 6;
 3 | 7 | 5 | 3 | 7   2+9+4=15;  8+7+9=24;
   |---+---+---|     6+7+2=15;  6+3+9=18;
```

```
 4 | 2 | 9 | 4 | 2   1+5+9=15;  1+7+4=12;
   '---'---'---'     8+3+4=15;  8+5+2=15;
* Complementary Pairs:
    6+4=10; 1+9=10; 8+2=10; 7+3=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[6]
   .---.---.---. Rows & Columns  Pan-diagonals
 2 | 6 | 7 | 2 | 6   6+7+2=15;  6+5+4=15;
   |---+---+---|     1+5+9=15;  7+9+8=24;
 9 | 1 | 5 | 9 | 1   8+3+4=15;  2+1+3= 6;
   |---+---+---|     6+1+8=15;  6+9+3=18;
 4 | 8 | 3 | 4 | 8   7+5+3=15;  7+1+4=12;
   '---'---'---'     2+9+4=15;  2+5+8=15;
* Complementary Pairs:
    6+4=10; 7+3=10; 2+8=10; 1+9=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[7]
   .---.---.---. Rows & Columns  Pan-diagonals
 6 | 8 | 1 | 6 | 8   8+1+6=15;  8+5+2=15;
   |---+---+---|     3+5+7=15;  1+7+4=12;
 7 | 3 | 5 | 7 | 3   4+9+2=15;  6+3+9=18;
   |---+---+---|     8+3+4=15;  8+7+9=24;
 2 | 4 | 9 | 2 | 4   1+5+9=15;  1+3+2= 6;
   '---'---'---'     6+7+2=15;  6+5+4=15;
* Complementary Pairs:
    8+2=10; 1+9=10; 6+4=10; 3+7=10; 5+5=10;
  ... This is a Self-complementary Magic Square!
[8]
   .---.---.---. Rows & Columns  Pan-diagonals
 4 | 8 | 3 | 4 | 8   8+3+4=15;  8+5+2=15;
   |---+---+---|     1+5+9=15;  3+9+6=18;
 9 | 1 | 5 | 9 | 1   6+7+2=15;  4+1+7=12;
   |---+---+---|     8+1+6=15;  8+9+7=24;
 2 | 6 | 7 | 2 | 6   3+5+7=15;  3+1+2= 6;
   '---'---'---'     4+9+2=15;  4+5+6=15;
* Complementary Pairs:
    8+2=10; 3+7=10; 4+6=10; 1+9=10; 5+5=10;
  ... This is a Self-complementary Magic Square!

 [Count = 8]
  OK!
```

As you see, the value of n5 always takes 5 and nothing else. 'Complementary Pairs of 10' such as {(1, 9), (2, 8), (3, 7), (4, 6), ...} are all located symmetrically with respect to the geometric center where the value 5 is placed.

Yes. All of them are certainly 'Self-Complementary' magic squares and nothing else, while we did not define them as they are at the first stage.

But why do they appear as self-complementary type? What makes it possible?

Let's have some algebraic study here for a while.

4. Let's get back to the first definition stage.

*** Basic Form and Simultaneous Equations ***

```
  .--.--.--.   n1+n2+n3=15 ... eq1;
  |n1|n2|n3|   n4+n5+n6=15 ... eq2;
  |--+--+--|   n7+n8+n9=15 ... eq3;
  |n4|n5|n6|   n1+n4+n7=15 ... eq4;
  |--+--+--|   n2+n5+n8=15 ... eq5;
  |n7|n8|n9|   n3+n6+n9=15 ... eq6;
  '--'--'--'   n1+n5+n9=15 ... eq7;
               n3+n5+n7=15 ... eq8;
eq7+eq8+eq2+eq5
  n1+n5+n9+n3+n5+n7+n4+n5+n6+n2+n5+n8=4*15
  n1+n2+n3+n4+n5+n5+n5+n5+n6+n7+n8+n9=60
    (n1+n2+n3+n4+n5+n6+n7+n8+n9)+3*n5=60
                    (1+9)*9/2    +3*n5=60
  Therefore                        n5=5  ... r1
Return this value to some equations above.
  eq2 means: n4+ 5+n6=15 -> n4+n6=10    ... r2
  eq5 means: n2+ 5+n8=15 -> n2+n8=10    ... r3
  eq7 means: n1+ 5+n9=15 -> n1+n9=10    ... r4
  eq8 means: n3+ 5+n7=15 -> n3+n7=10    ... r5
They mean  {(n1,n9),(n2,n8),(n3,n7),(n4,n6)}={(1,9),(2,8),(3,7),(4,6)}
```

All 'Self-Complementary Conditions' are logically derived from our first definitions. This can explain the reason why they are always Self-complementary type.

5. What else could you notice about those eight solutions?

The value '1' never comes to n1, n3, n7 or n9, on any top of the four corners.

Let's try to prove this by 'reductio ad absurdum'.

```
If n1=1, then n9=9 ... r4';
  and n2+n3=14 ... eq1'; n4+n7=14 ... eq4';
  eq1'+eq4'  n2+n3+n4+n7=28;
  but n3+n7=10 ... r5; therefore n2+n4=18;
  It means (n2,n4)={(1,17), (2,16), (3,15), ..., (8,10) or (9,9)}
```

All those pairs cannot be combined, because we can only use each of {1, 2, 3, 4, ..., 8 or 9} strictly once. We cannot use any number over 9, and we must not use 9 twice for any position, either.

We must give up the first supposition 'n1=1'. It is impossible in any logical sense. We also must give up any hope of supposing 'n3=1', 'n7=1' or 'n9=1', which you can move to the position n1 by simple rotations by 90, 180 or 270 degrees.

6. Could you notice those eight solutions are all the same, eight different 'faces' of the same body? Examine each content of 3 rows, 3 columns and 2 diagonals.

They differ only in their positions, but they are all the same in their contents.

Each one must be classified as a reflected pattern of any other form or as a rotated pattern by 90, 180 or 270 degrees of any other one.

We want to count one for the only body instead of those eight different 'faces'.

What should we do in order to print out the single solution?

We may well use the 'List-forming Inequality Conditions' for that purpose.

Won't you see the next demonstration of selecting it by such conditions?

```
** Composing Magic Squares of Order 3: **
** The Effect of Inequality Conditions **
```

```
* Basic Form     n1+n2+n3=C;    n1+n4+n7=C;
  n1 n2 n3       n4+n5+n6=C;    n2+n5+n8=C;
  n4 n5 n6       n7+n8+n9=C;    n3+n6+n9=C;
  n7 n8 n9       n1+n5+n9=C;    n3+n5+n7=C;
      ('C' means the Magic Constant 15.)

* Without any Inequality Condition *
        1/          2/          3/          4/
   2  7  6     2  9  4     4  3  8     4  9  2
   9  5  1     7  5  3     9  5  1     3  5  7
   4  3  8     6  1  8     2  7  6     8  1  6
        5/          6/          7/          8/
   6  1  8     6  7  2     8  1  6     8  3  4
   7  5  3     1  5  9     3  5  7     1  5  9
   2  9  4     8  3  4     4  9  2     6  7  2
 [Count = 8]

* Under n3<n7; *
        1/          2/          3/          4/
   2  9  4     4  9  2     6  7  2     8  3  4
   7  5  3     3  5  7     1  5  9     1  5  9
   6  1  8     8  1  6     8  3  4     6  7  2
 [Count = 4]

* Under n1<n9; *
        1/          2/          3/          4/
   2  7  6     2  9  4     4  3  8     4  9  2
   9  5  1     7  5  3     9  5  1     3  5  7
   4  3  8     6  1  8     2  7  6     8  1  6
 [Count = 4]

* Under {n1<n3; n1<n7; n1<n9;} *
        1/          2/
   2  7  6     2  9  4
   9  5  1     7  5  3
   4  3  8     6  1  8
 [Count = 2]

* Under {n1<n3; n1<n7; n1<n9; n3<n7;} *
        1/
   2  9  4
   7  5  3
   6  1  8
 [Count = 1]  OK!
```

You may well call the 'Standard Solution' of MS33 for the last one selected, because it could be the representative solution for all the eight primitive ones.

Section 2. Standard Solutions of Self-Complementary Magic Squares 3x3

1. Now that we can only make one type of solutions, we may well define our object as a 'self-complementary' magic square of order 3 from the beginning.

Let's design such an object and make the standard solution of 'self-complementary'

magic squares of order 3 now.

The next program list shows a sample of my recent dictation.

```
/** Sophisticated PROGRAM #2 for **/
/** Self-Complementary Magic Squares 3x3 **/
/**  'MS33SC2.c' built by Kanji Setsuda  **/
/**  Dec.24, 2005; on MacOSX & Xcode 1.5 **/
/**/
#include <stdio.h>
/**/
short cnt, cnt2;
short LSM, CC;
short nm[10], uflg[10];
short tn[5][10];
/**/
void stp1(void), stp2(void), stp3(void);
void stp4(void), stp5(void), stp6(void);
void stp7(void), stp8(void), stp9(void);
void recordans(void);
void prans(short x);
/**/
/* Main Program */
int main(){
 short n;
  printf("\n** Self-Complementary Magic Squares of Order 3 **\n");
  for(n=0;n<10;n++){nm[n]=0; uflg[n]=0;}
  LSM=15; CC=10; nm[5]=5;
  cnt=0; cnt2=0;
  stp1(); /* Calculations */
  if(cnt2>0){prans(cnt2);}
  printf(" [Count = %d]\n",cnt);
  printf("  OK!\n");
 return 0;
}
/* Calculations */
/* Set n2 and n8 */
void stp1(){
 short m,n;
  for(n=1;n<10;n++){m=CC-n;
    if((uflg[n]==0)&&(uflg[m]==0)){
      nm[2]=n; nm[8]=m;
      uflg[n]=1; uflg[m]=1;
      stp2();
      uflg[m]=0; uflg[n]=0;
    }
  }
}
/**/
/* Set n4 & n6 */
void stp2(){
 short m,n;
  for(n=1;n<10;n++){m=CC-n;
    if((uflg[n]==0)&&(uflg[m]==0)){
      nm[4]=n; nm[6]=m;
      uflg[n]=1; uflg[m]=1;
      stp3();
      uflg[m]=0; uflg[n]=0;
    }
  }
```

```
}
/**/
/* Set n1 & n9 */
void stp3(){
 short m,n;
  for(n=1;n<10;n++){m=CC-n;
    if((uflg[n]==0)&&(uflg[m]==0)){
      nm[1]=n; nm[9]=m;
      uflg[n]=1; uflg[m]=1;
      stp4();
      uflg[m]=0; uflg[n]=0;
    }
  }
}
/**/
/* Set n3=LSM-n1-n2 & n3=LSM-n6-n9 & n7 */
void stp4(){
 short m,n;
  m=LSM-nm[1]-nm[2];
  n=LSM-nm[6]-nm[9];
  if((0<m)&&(m<10)&&(m==n)){n=CC-m;
    if((uflg[m]==0)&&(uflg[n]==0)){
      nm[3]=m; nm[7]=n;
      uflg[m]=1; uflg[n]=1;
      stp5();
      uflg[n]=0; uflg[m]=0;
    }
  }
}
/**/
/* Check the Line Sums */
void stp5(){
 short csm1,csm2,csm3;
  csm1=nm[1]+nm[4]+nm[7];
  csm2=nm[7]+nm[8]+nm[9];
  csm3=nm[1]+nm[5]+nm[9];
  if((csm1==LSM)&&(csm2==LSM)&&(csm3==LSM)){recordans();}
}
/**/
/* Record the Answer to the Table */
void recordans(){
 short m,n;
  tn[cnt2][0]=cnt+1;
  for(n=1;n<10;n++){tn[cnt2][n]=nm[n];}
  cnt++; cnt2++; m=4;
  if(cnt2==m){prans(m); cnt2=0;}
}
/**/
/* Print the Answers */
void prans(short x){
 short l,l3,n;
  for(n=0;n<x;n++){printf("%14d/",tn[n][0]);}; printf("\n");
  for(n=0;n<x;n++){printf("  .---.---.---.");}; printf("\n");
  for(l=0;l<3;l++){l3=l*3;
   for(n=0;n<x;n++){
      printf("  |%2d |%2d |%2d |",tn[n][l3+1],tn[n][l3+2],tn[n][l3+3]);}
    printf("\n");
  if(l<2){for(n=0;n<x;n++){printf("  |---+---+---|");}; printf("\n");}
  }
```

```
  for(n=0;n<x;n++){printf("  '---'---'---'");}; printf("\n");
}
/**/
```

This program should print out all the eight 'primitive solutions' as follows.

```
** Self-Complementary Magic Squares of Order 3 **
            1/            2/            3/            4/
  .---.---.---.  .---.---.---.  .---.---.---.  .---.---.---.
  | 8 | 1 | 6 |  | 6 | 1 | 8 |  | 8 | 3 | 4 |  | 4 | 3 | 8 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 3 | 5 | 7 |  | 7 | 5 | 3 |  | 1 | 5 | 9 |  | 9 | 5 | 1 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 4 | 9 | 2 |  | 2 | 9 | 4 |  | 6 | 7 | 2 |  | 2 | 7 | 6 |
  '---'---'---'  '---'---'---'  '---'---'---'  '---'---'---'
            5/            6/            7/            8/
  .---.---.---.  .---.---.---.  .---.---.---.  .---.---.---.
  | 6 | 7 | 2 |  | 2 | 7 | 6 |  | 4 | 9 | 2 |  | 2 | 9 | 4 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 1 | 5 | 9 |  | 9 | 5 | 1 |  | 3 | 5 | 7 |  | 7 | 5 | 3 |
  |---+---+---|  |---+---+---|  |---+---+---|  |---+---+---|
  | 8 | 3 | 4 |  | 4 | 3 | 8 |  | 8 | 1 | 6 |  | 6 | 1 | 8 |
  '---'---'---'  '---'---'---'  '---'---'---'  '---'---'---'
 [Count = 8]  OK!
```

I took the different style here from the previous program. It is because we know the number '1' never comes to the variable n1, and we don't have to determine n1 first of all. We should rather prefer to determine n2 first, so that '1' should come here as early as possible. We should determine the 2nd column first and the 2nd row next.

How do we have to arrange the list-forming inequality conditions for this case, when we want to have the 'standard solution' of them?

{n2<n4; n2<n6; n2<n8; and n4<n6;} is my answer.

Modify the program above just a little as follows.

```
/* Calculations */
/* Set n2 and n8 & n2<n8 */
void stp1(){
 short m,n;
  for(n=1;n<5;n++){m=CC-n;
    if((uflg[n]==0)&&(uflg[m]==0)){
  . . . . .
}
/**/
/* Set n4 & n6 & n2<n4<n6 */
void stp2(){
 short m,n;
  for(n=nm[2]+1;n<5;n++){m=CC-n;
    if((uflg[n]==0)&&(uflg[m]==0)){
  . . . . .
  }
}
/**/
```

With these modifications you should have got the only one 'standard solution' of our object.

```
** List of the Only One Standard Solution **
  ** (n2<n4; n2<n6; n2<n8; and n4<n6) **
```

```
          1/
.---.---.---.
| 8 | 1 | 6 |
|---+---+---|
| 3 | 5 | 7 |
|---+---+---|
| 4 | 9 | 2 |
'---'---'---'    [Count = 1]
```

2. What kind of relations can we find between the 'primitive' eight solutions and the 'standard' one? What makes the eight from the only one?

If you put those eight in such an order as follows, you can easily know the relations.

```
      1/  ->     4/  ->     8/  ->     5/ (Rotation)
  8  1  6    4  3  8    2  9  4    6  7  2
  3  5  7    9  5  1    7  5  3    1  5  9
  4  9  2    2  7  6    6  1  8    8  3  4
    | (Reflection)
    V  3/  ->     2/  ->     6/  ->     7/ (Rotation)
  8  3  4    6  1  8    2  7  6    4  9  2
  1  5  9    7  5  3    9  5  1    3  5  7
  6  7  2    2  9  4    4  3  8    8  1  6
```

Or you can put them in such a different order as shown below.

```
      1/  ->     4/  ->     8/  ->     5/ (Rotation)
  8  1  6    4  3  8    2  9  4    6  7  2
  3  5  7    9  5  1    7  5  3    1  5  9
  4  9  2    2  7  6    6  1  8    8  3  4
    |          |          |          |     (Reflection)
    V  3/      V  7/      V  6/      V  2/
  8  3  4    4  9  2    2  7  6    6  1  8
  1  5  9    3  5  7    9  5  1    7  5  3
  6  7  2    8  1  6    4  3  8    2  9  4
```

It can be explained by the two types of transformations: (1) Rotation by 90 degrees clockwise and (2) mirror reflection with respect to the primary diagonal n1+n5+n9.

You don't have to take a rotation by 180 degrees or by 270 degrees, for you can do the same thing by repeating the rotation by 90 degrees clockwise two or three times.

3. Let's reconstruct all the 'primitive' eight solutions from the only 'standard' one using those two types of transformation: rotation and reflection.

Add the next procedures to the previous program and call 'trnsf();' from the main body, and you should get the list of eight primitive solutions transformed.

```
* Concept Diagrams for Transformation *
        1/  ->     2/  ->     3/  ->     4/ (Rotation)
  n1 n2 n3    7  4  1    9  8  7    3  6  9
  n4 n5 n6    8  5  2    6  5  4    2  5  8
  n7 n8 n9    9  6  3    3  2  1    1  4  7
     |          |          |          |     (Reflection)
     V  5/      V  6/      V  7/      V  8/
   1  4  7    7  8  9    9  6  3    3  2  1
   2  5  8    4  5  6    8  5  2    6  5  4
   3  6  9    1  2  3    7  4  1    9  8  7
```

```
/**/
/* Mirror Reflection */
void reflct(){
 short n;
  tnm[1]=nm[1]; tnm[2]=nm[4]; tnm[3]=nm[7];
  tnm[4]=nm[2]; tnm[5]=nm[5]; tnm[6]=nm[8];
  tnm[7]=nm[3]; tnm[8]=nm[6]; tnm[9]=nm[9];
  for(n=1;n<10;n++){nm[n]=tnm[n];}
}
/* Rotation by 90 degrees */
void rotat90(){
 short n;
  tnm[1]=nm[7]; tnm[2]=nm[4]; tnm[3]=nm[1];
  tnm[4]=nm[8]; tnm[5]=nm[5]; tnm[6]=nm[2];
  tnm[7]=nm[9]; tnm[8]=nm[6]; tnm[9]=nm[3];
  for(n=1;n<10;n++){nm[n]=tnm[n];}
}
/**/
/* Transform One into Eight */
void trnsf (){
 short m,n;
  for(n=1;n<10;n++){nm[n]=tans[0][n];}
  for(m=1;m<4;m++){
    rotat90();
    for(n=1;n<10;n++){tans[m][n]=tnm[n];}
    tans[m][0]=m+1;
  }
  for(m=4;m<8;m++){
    for(n=1;n<10;n++){nm[n]=tans[m-4][n];}
    reflct();
    for(n=1;n<10;n++){tans[m][n]=tnm[n];}
    tans[m][0]=m+1;
  }
}
/**/
```

```
* Transform 1 into 8: by Reflection and Rotation *
          1/          2/          3/          4/
    8  1  6     4  3  8     2  9  4     6  7  2
    3  5  7     9  5  1     7  5  3     1  5  9
    4  9  2     2  7  6     6  1  8     8  3  4
          5/          6/          7/          8/
    8  3  4     4  9  2     2  7  6     6  1  8
    1  5  9     3  5  7     9  5  1     7  5  3
    6  7  2     8  1  6     4  3  8     2  9  4
 [Count = 8]  OK!
```

No.1 can be nominated for the representative solution of all. You don't have to have all those eight in your mind any longer, or save in your outer memory device, either.

Thus, our transformation system really acts in the two ways:

(1) It classifies the primitive eight into one group whose representative is called as the 'standard' solution.

(2) It reconstructs all the primitive eight from the only 'standard' one.

It is always true in whatever order and in whatever type you may take for your two-dimensional magic squares.

Section 3. Pan-diagonal Magic Squares of Order 3

1. You may probably know about the concept of pan-diagonal magic squares, but let me explain a little about it right here.

When you move any first(or last) row or column far to the opposite side and put it beyond the last(or first) one, you can make some new squares as follows. You can even repeat the same thing to the result and make another set of new squares. All new diagonals in those squares must always add up to the magic constant 15. It is the core concept of any pan-magic types.

```
* Concept of Pan-diagonal Magic Squares *
                      N1/
                 .--.--.--.
 /Original  .->|n7|n8|n9|                     N3/         N4/  ...
 .--.--.--. |  |--+--+--|              .--.--.--.  .--.--.--.
 |n1|n2|n3|-+. |n1|n2|n3|         N2/  |n2|n3|n1|  |n3|n1|n2|
 |--+--+--| || |--+--+--|  .--.--.--.  |--+--+--|  |--+--+--|
 |n4|n5|n6| || |n4|n5|n6|  |n4|n5|n6|  |n5|n6|n4|  |n6|n4|n5|
 |--+--+--| || '--'--'--'  |--+--+--|  |--+--+--|  |--+--+--|
 |n7|n8|n9|-'|             |n7|n8|n9|  |n8|n9|n7|  |n9|n7|n8|
 '--'--'--'  |             |--+--+--|  '--'--'--'  '--'--'--'
             '------------>|n1|n2|n3|
                           '--'--'--'
 New diagonals must always add up to 15.
             n7+n2+n6=15; n4+n8+n3=15; n2+n6+n7=15; n3+n4+n8=15; ...
             n9+n2+n4=15; n6+n8+n1=15; n1+n6+n8=15; n2+n4+n9=15; ...
```

How many new squares can you make in all by such shifting rows and columns?
How many different diagonals do you have to make their sums equal to 15?
You can make 9 new squares, and you must have 6 new diagonals add up to 15.
We usually call these 6 new diagonals as 'Pan-diagonals' of the original.

2. I would define this concept of pan-magic type at the first stage in the next way as follows, with the basic forms and basic simultaneous equations:

```
** Pan-diagonal Magic Square of Order 3 **
* Basic Form and Basic Equations *
                 .--.--.--.
  /Classic    n9|n7 n8 n9|n7 n8
 .--.--.--.     .--.--.--.--.
 |n1|n2|n3|   n3|n1|n2|n3|n1|n2
 |--+--+--|     |--+--+--| -|
 |n4|n5|n6|   n6|n4|n5|n6|n4|n5
 |--+--+--|     |--+--+--| -|
 |n7|n8|n9|   n9|n7|n8|n9|n7|n8
 '--'--'--'     '--'--'--'--'
               n3|n1 n2 n3|n1 n2
                 '--'--'--'

* Rows and Columns:      * Pan-diagonals:
 n1+n2+n3 = 15 ... rw1;  n1+n5+n9 = 15 ... pd1
 n4+n5+n6 = 15 ... rw2;  n2+n6+n7 = 15 ... pd2
 n7+n8+n9 = 15 ... rw3;  n3+n4+n8 = 15 ... pd3
 n1+n4+n7 = 15 ... cl1;  n1+n6+n8 = 15 ... pb1
 n2+n5+n8 = 15 ... cl2;  n2+n4+n9 = 15 ... pb2
 n3+n6+n9 = 15 ... cl3;  n3+n5+n7 = 15 ... pb3
```

How many pan-magic squares in all can we make with these conditions, then?

Nothing! I am sorry to say that, but it is really true we cannot make any pan-magic squares of order 3 at all.

3. Let's try to prove this by the 'reductio ad absurdum' here.

Divide all cases into three ones as below and make the value test for each:

```
* Case #1:
If n1=1 then [n2+n3=n4+n7=n5+n9=n6+n8=14;
  It means:{(n2,n3),(n4,n7),(n5,n9),(n6,n8)}
           ={(5,9),(6,8),(7,7),(8,6),(9,5)};
  ]
* Case #2:
If n2=1 then [n1+n3=n5+n8=n6+n7=n4+n9=14;
  It means:{(n1,n3),(n5,n8),(n6,n7),(n4,n9)}
           ={(5,9),(6,8),(7,7),(8,6),(9,5)};
  ]
* Case #3:
If n5=1 then [n4+n6=n2+n8=n1+n9=n3+n7=14;
  It means:{(n1,n9),(n2,n8),(n3,n7),(n4,n6)}
           ={(5,9),(6,8),(7,7),(8,6),(9,5)};
  ]
```

But in every case we can only make two valid Complementary Pairs of 14 and we cannot satisfy all the equations above. It is because we cannot use any number twice or more often, and cannot use any number over 9, either.

It means we must give up our supposition (n1=1), (n2=1) or (n5=1).

But can we compose any pan-magic squares without (n1=1), (n2=1) or (n5=1)?

No! Because (n3=1), (n7=1), (n9=1), (n4=1), (n6=1) and (n8=1) are also impossible. They mean all the same to (n1=1) or (n2=1) in any rotated patterns of the original.

We cannot use '1' in any position at last. That is definitely against the first promise.

Since these contradictions should come under the first definition of pan-magic type, we must give it up. We cannot compose any pan-diagonal magic squares of order 3.

(Original written in Japanese on May 26, 2000;
Revised and written in English on Dec.27, 2005;
by Kanji Setsuda; Worked on MacOSX and Xcode 1.5)

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