Research Open Journal of Information Science and Application
Vol. 1, No. 1, August 2013, PP:01-19
Available online at http://www.scitecpub.com/Journals.php

Research article

Special Magic Squares of Order Six

S. Al-Ashhab

University of Dammam (KSA)

E-mail: saleemashhab1@yahoo.com

Abstract

In this paper we introduce and study special types of magic squares of order six. We list some enumerations of these squares. We present a parallelizable code. This code is based on the principles of genetic algorithms.

Keywords: Magic Squares, Four Corner Property, Parallel Computing, Search Algorithms, Nested loops.

1 Introduction

A magic square is a square matrix, where the sum of all entries in each row or column and both main diagonals yields the same number. This number is called the magic constant. A natural magic square of order n is a matrix of size n×n such that its entries consists of all integers from one to n^2. The magic constant in this case is . A symmetric magic square is a natural magic square of order n such that the sum of all opposite entries equals n+1. For example,

Table 1: a natural symmetric magic square

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

A pandiagonal magic square is a magic square such that the sum of all entries in all broken diagonals equals the magic constant. For example, we note in table 2 that the sum of the entries 39,12,46,22,20,23,13 is 175, which is the magic sum. These entries represent the first right broken diagonal.

Table 2: a natural pandiagonal and symmetric magic square of order seven

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

In the seventeenth century Frenicle de Bessy claimed that the number of the 4x4 magic squares is 880, where he considered a magic square with all its reflections and rotations one square. Hire listed them all in a table in the year 1693. Recently we can use the computer to check that there are

$$880*8 = 7040$$

magic squares of order 4.

In 1973 the number of all natural magic squares of order five became known. Schoeppel computed it using a PDF-10 machine. It is

$$64\ 826\ 306*32=2\ 202\ 441\ 792$$

where we multiply with 32 due the existence of type preserving transformations. According to [5] there exists

$$736\ 347\ 893\ 760$$

natural nested magic squares of order six.

It is well-known that there are pandiagonal magic squares and symmetric squares of order five. But, there are neither pandiagonal magic squares nor symmetric squares of order six. The number of natural magic squares of order six is actually till now unknown. Trump made using statistical methods (Monte Carlo Backtracking) the following interval estimation for this number

$$(1.7712e19,\ 1.7796e19)$$

with a probability of 99%. We give here the number of a subset of such squares. We define here classes of magic squares of order six, which satisfy some of the conditions for both types.

The most-perfect pandiagonal magic squares of McClintock (cf. [11]) for which Ollerenshaw and Brée's (cf. [10]) combinatorial count ranks as a major achievement, draw attention to another type which have the same sum for all 2 by 2 subsquares (or quartets). The number of complete magic squares of order four is 48, and the number of complete magic squares of order eight (cf. [10]) is

368 640.

Ollerenshaw and Brée (cf. [10]) have a patent for using most-perfect magic squares for cryptography, and Besslich (cf. [7] and [8]) has proposed using pandiagonal magic squares as dither matrices for image processing.

A pandiagonal and symmetric magic square is called ultramagic. According to [14] the number of ultramagic squares of order five is 16 and number of ultramagic squares of order seven is

20 190 684.

The weakest property of a square is being semi magic. By this concept we mean a matrix, where the sum of all entries in each row or column yields the magic constant. According to Trump (cf. [14]) the number of semi magic squares of order four is

68 688,

and the number of semi magic squares of order five is

579 043 051 200.

2 Four corner magic square

This concept was first introduced in [1]. Alashhab studied this type there in very simple cases. In [1] Al-ashhab considered the typecalled nested four corner magic square with a pandiagonal magic square. We continue here the study of this type.

We focus in this paper on the following kind of magic squares: magic squares of order six with magic constant 3s such that

$$a_{ij} + a_{(i+3)(j+3)} + a_{i(j+3)} + a_{(i+3)j} = 2s$$

holds for each i=1,2,3 and j=1,2,3 and

$$a_{33} + a_{44} + a_{34} + a_{43} = 2s.$$

We call such squares four corner magic square of order 6. The entries of a four corner magic square of order 6 satisfy

$$a_{14}+a_{25}+a_{36}+a_{41}+a_{52}+a_{63}=3s,\ a_{13}+a_{22}+a_{31}+a_{61}+a_{55}+a_{64}=3s$$

These two conditions represent the sum of the entries of two broken diagonals. If the magic square is pandiagonal, then we have to consider all broken diagonals. To see the validity of the first equation we know from the definition that

$$a_{11}+a_{44}+a_{14}+a_{41}=2s,\ a_{22}+a_{55}+a_{25}+a_{52}=2s,\ a_{33}+a_{66}+a_{36}+a_{63}=2s$$

holds. Adding up these equations and subtracting from them the following equation

$$a_{11}+a_{22}+a_{33}+a_{44}+a_{55}+a_{66}=3s$$

yields the desired equation.

A four corner magic square of order 6 can be written as

Table 3: a symbolic four corner magic squares

x	f	G	t	M	G
z	h	N	j	q	N
w	E	e	a	m	D
A	k	2s – a – b – e	b	H	R
2s – j – o – z	p	d	o	2s – p – q – h	T
B	F	W	J	L	Y

where

$$A=2s-b-t-x,$$

$$B=j+o+t+b-s-w,$$

$$D=d+g+n+x-a-p-q,$$

$$E=3s-a-e-m-w-D,$$

$$F=3s-f-h-k-p-E,$$

$$G=2s+e+w-(j+o+p+q+t),$$

$$H=e+g+s+w+x-j-k-o-p-q,$$

$$J=3s-j-b-o-a-t,$$

$$M=3s-f-g-t-x-G,$$

$$N=3s - h - j - n - q - z,$$

$$L= f + h + k + p - m - s,$$

$$R = a + b + j + o + p + q + t - g - 2s - w,$$

$$T= h + j + q + z - d - s,$$

$$W=a + b + s - d - g - n,$$

$$Y=s + p + q - b - e - x.$$

We see that it has seventeen independent variables, which are represented by the small letters. In the code these variables will be assigned to loops. We have for example

Table 4: a natural four corner magic squares

6	23	11	13	33	25
19	28	36	3	7	18
2	29	1	17	27	35
21	8	22	34	10	16
32	9	15	20	30	5
31	14	26	24	4	12

2.1 Four corner magic square with positive center.

We introduce now the main concept in our work. We call a four corner magic squares such that

$$a_{33} * a_{44} - a_{34} * a_{43} > 0$$

a four corner magic square of order six with positive center. This means that the 2 by 2 square in the center has positive determinant.

2.1. Property preserving transformations

There are seven classical transformations, which take a magic square into another magic square. They are the combinations of the rotations with angles $\pi/2$, π, $(3\pi)/2$ and transpose operation. Now, a four corner magic squares with positive center can be transformed as follows into another one of the same kind: we make these interchanges simultaneously: interchange a_{12} (res. a_{62}) with a_{15} (res. a_{65}), interchange a_{21} (res. a_{26}) with a_{51} (res. a_{56}), interchange a_{22} (res. a_{55}) with a_{25} (res. a_{52}), interchange a_{23} (res. a_{24}) with a_{53} (res. a_{54}), interchange a_{32} (res. a_{42}) with a_{35} (res. a_{45}).

It is obvious that the center remains unchanged by this transformation.

We can use this transformation to reduce the number of computed natural magic squares. In order to eliminate the effect of the previous transformations we compute all natural four corner magic squares with positive center for which the following conditions hold:

$$p < q,\ a < e < b,\ a < 2s - a - b - e.$$

This means that we compute first the number of all natural squares satisfying these conditions.We multiply then the number with sixteen in order to get the number of squares.

2.2 Number of squares

We used computers to count several types of magic squares. The algorithm is constructed in such a way that we take specific values at the beginning. In the case of four corner magic squares with positive center we fix by each run of the code two specific values for a, b and e, which satisfy the following conditions

$$a < e < b,\ a < 2s - a - b - e,$$

$$b*e-a*(2s - a - b - e) > 0.$$

We list the number for all squares with respect to different values of a, b and e in the following tables:

Table 5: a list of the number of four corner magic squares with a=5

b	e	number	B	e	number
28	6	142275478	27	6	151881687

Table 6: a list of the number of four corner magic squares with a=6

B	e	number	B	e	number
25	7	148879623	24	8	151153974
26	7	145560084	25	8	141632641
27	7	142830575	23	9	143135768
28	7	142272721			

Table 7: a list of the number of four corner magic squares with a=7

B	e	number	b	e	number	b	e	number
23	8	149240219	24	9	134990744	19	12	134502101
24	8	148615081	25	9	138418197	20	12	140868624
25	8	141213764	21	10	142662660	18	13	135435276
26	8	140975397	22	10	143067910	17	14	133747793
27	8	139363210	23	10	142163907	16	15	131305331
22	9	145154107	20	11	135733078			
23	9	152800164	21	11	144846545			

Table 8: a list of the number of four corner magic squares with a=8

B	e	number	b	e	number	b	e	number
21	9	143594872	19	11	134376506	17	13	135682553
22	9	144389905	20	11	150732635	18	13	129286147
23	9	139990551	21	11	135622044	19	13	129499341
24	9	137756276	22	11	133406533	20	13	132563331
25	9	132542832	23	11	143270620	16	14	137623014
26	9	132604030	18	12	142888748	17	14	141274205
20	10	145372240	19	12	127643145	18	14	132363823
21	10	134488912	20	12	142521759	16	15	128022232
22	10	143940250	21	12	139970728	17	15	128384420
23	10	130550812						
24	10	139231388						

Table 9: a list of the number of four corner magic squares with a=9

b	e	number	b	e	number	b	e	number
19	10	149173141	22	11	133571878	19	13	129942329
20	10	138342068	23	11	131190172	20	13	130317255
21	10	138917234	24	11	136686393	21	13	127300394
22	10	137244633	17	12	135810466	15	14	134121525
23	10	143761353	18	12	132200077	16	14	137809464
24	10	134848602	19	12	133747474	17	14	131063343
25	10	132292873	20	12	136645500	18	14	130230377
26	10	141770813	21	12	131887930	19	14	126767780
18	11	138190635	22	12	133190481	16	15	128899433
19	11	139701915	16	13	134603089	17	15	121418695
20	11	133867159	17	13	135108324	18	15	119459984
21	11	135320277	18	13	127926756	17	16	121120361

Table 10: a list of the number of four corner magic squares with a=10

B	e	number	B	e	number	b	e	number
17	11	137800262	19	12	126840267	15	14	131797562
18	11	136432511	20	12	132626465	16	14	131031874
19	11	134995272	21	12	127958340	17	14	124066378
20	11	134093384	22	12	128229502	18	14	125843075
21	11	142075092	23	12	132054199	19	14	122625761
22	11	132569706	15	13	138687369	20	14	119715913
23	11	130730850	16	13	136013377	16	15	130665900

24	11	134847246	17	13	140330105	17	15	126281250
25	11	145534400	18	13	127023594	18	15	119744242
16	12	143478320	19	13	123451990	19	15	118056469
17	12	132397335	20	13	134963983	17	16	118375804
18	12	135608506	21	13	128316619	18	16	118193688
			22	13	121958143			

Table 11: a list of the number of four corner magic squares with a=11

B	e	number	B	e	number	b	e	Number
15	12	136855015	14	13	142088500	19	14	121366787
16	12	134760196	15	13	135246322	20	14	130615818
17	12	132847711	16	13	131115942	21	14	121807251
18	12	133569930	17	13	124754709	16	15	121839807
19	12	136352497	18	13	122768564	17	15	119458731
20	12	126870022	19	13	125854607	18	15	125037058
21	12	126432333	20	13	120619213	19	15	118971662
22	12	123942478	21	13	125776194	20	15	110774224
23	12	126558532	22	13	123377288	17	16	119672738
24	12	134159325	15	14	131447556	18	16	115067660
			16	14	129648211	19	16	114191892
			17	14	135611220	18	17	118351247
			18	14	122474344			

Table 12: a list of the number of four corner magic squares with a=12

B	E	number	b	e	number	b	e	Number
14	13	141617171	23	13	135746592	16	15	118468772
15	13	134264186	15	14	128764559	17	15	127438703
16	13	131439570	16	14	126599940	18	15	118791505
17	13	126275986	17	14	119401141	19	15	121100417
18	13	129087754	18	14	122986578	20	15	117601428
19	13	129410433	19	14	119205291	17	16	121499468
20	13	125161145	20	14	114746404	18	16	116805362
21	13	122029111	21	14	118858318	19	16	113047908
22	13	125597483	22	14	120034009	18	17	116497767

Table 13: a list of the number of four corner magic squares with a=13

B	e	number	b	e	number	b	e	number
15	14	130498831	22	14	119960980	17	16	122373478
16	14	125313274	16	15	124866991	18	16	121594118

17	14	123980424	17	15	120442628	19	16	118305846
18	14	123890374	18	15	119115855	20	16	124468426
19	14	124584159	19	15	114368512	18	17	117069993
20	14	119898468	20	15	123533289	19	17	115335036
21	14	118922094	21	15	115808535			

Table 14: a list of the number of four corner magic squares with a=14

b	e	number	b	e	number
16	15	125477566	17	16	124594623
17	15	126616800	18	16	119948160
18	15	120465231	19	16	121813151
19	15	121716664	20	16	121004372
20	15	122966558	18	17	127042593
21	15	129832011	19	17	124677774

Table 15:a list of the number of four corner magic squares with a=15

b	e	number	b	e	number
17	16	124970823	20	16	132348932
18	16	132696180	18	17	138772449
19	16	129786974	19	17	127552211

Table 16: a list of the number of four corner magic squares with a=16

b	e	number	b	e	number
18	17	159355574	19	17	147854836

The total number of the squares is

$$30350772825.$$

Hence, there are

$$30350772825*16=485\ 612\ 365\ 200$$

different four corner magic squares of order six with positive center.

4 The number of four corner magic squares of order 6

The number of all different possible values for a, b and e by computing the number of four corner magic squares is 3429. Hence, there are 3429 possible centers of the natural four corner magic squares. The number of squares with positive center is (as illustrated) 232. The remaining squares include the squares with symmetric centers (cf. [2]) and semi symmetric centers(cf. [3]).

There are 153 possible symmetric centers of the natural four corner magic squares. According to [2] there are

28 634 584 244*16=458 153 347 904

different natural four corner magic squares with symmetric center.There are 306 possible semi symmetric centers of the natural four corner magic squares. According to [3] there are

101 425 060 998*16=1 622 800 975 968

different natural four corner magic squares with semi symmetric center.

Based on the information about the computed natural four corner magic squares we have considered

153+306+232=691

centers.Their total number is

30 215 164 319+,,,,,,,*16

We want here to estimate the number of four corner magic squares of order 6. By computing the number

$$\frac{2566566689072}{691} * 4329 = 16079113164967.7$$

Hence we estimate the number of the four corner magic squares to be

1.608e13

5 Parallelization and grid computing

The problem itself is split into several “part” problems, since counting squares for each center is a separate problem. The code is constructed so that the input is the center of the square. This is the first step by splitting the job of counting into many smaller jobs, which run in parallel. Since we can fix the value of the outer for-loop before running the code. By this way we can split the task into 36 tasks, which can run in parallel.

The C code is presented in the appendix. The code is parallelizable. The code (algorithm) uses nested for loops representing the independent variables (the small letters). The first loop is for the variable t. When we fix one center we also run the code for an interval of the values of the variable t. This interval is a part of the input. We have the freedom to choose any subinterval of [0,36]. By choosing smaller intervals we run smaller jobs since they do not involve much computations.

The loops are used to assign all possible values for these variables between 1 and 36. When we make a specific assignment for the independent variables, we substitute in the formulas, which are written in the definition. This determines a numerical matrix, which is then examined to be a possible magic square or not, i. e. the computed value for being in the range from 1 to 36 and for being different from other existing values.The computation were done with the aid of the EUMED GRID system. The jobs were submitted to the system, which distributes the jobs on the connected computers.

6 Conclusions

We have introduced several types of magic squares. The problem of counting these squares is not completely solved yet. We can find some numbers and estimations in [14]. The development of computers can help by this task. In this paper we presented some counting and ideas how to count. In the future this research can be extended to include more types and give counting for the introduced types. The code, which we presented, is based on the idea of search over all possibilities in such a way that we continue the search at each dead end from the nearest exit.

7 PROGRAM CODE (The C-code)

```
#include <assert.h>

#include <errno.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

constint N = 6;

constint NN = N*N;

constint Sum2 = NN - 1;         /*  35 */

constint Sum4 = Sum2 + Sum2;     /*  70 */

constintMsum = Sum2 + Sum4;     /* 105 */

structbools {  bool used[NN];};

structboolsallFree;

#define Uint unsigned int

voidwriteSquare(int *p, FILE *wfp)

{  charsquareString[120], *s = squareString; int cells = 0;

{int i; for (i = 0; i < NN; ++i) {int x = p[i] + 1;

if (x < 10)      { *s++ = ' '; *s++ = '0' + x; }

else if (x < 20) { *s++ = '1'; *s++ = '0' - 10 + x; }

else if (x < 30) { *s++ = '2'; *s++ = '0' - 20 + x; }
```

```
else            { *s++ = '3'; *s++ = '0' - 30 + x; }

if (++cells == N)  { *s++ = '\n'; cells = 0; } else *s++ = ' ';  }}

*s++ = '\n'; *s++ = '\0';

fputs(squareString, wfp);}

/* writeSquare */

UintmakeSquares(int a, int b, int e, int t0, int t1, int J, FILE *wfp)

{ Uint count = 0, pcount = 0;  bools v = allFree; int Z[NN]; Z[20]=J;

v.used[e] = true; v.used[a] = true; v.used[b] = true; v.used[J] = true;

for (int t = t0; t < t1; ++t) if (!v.used[t]) {    v.used[t] = true;

for (int x = 0; x < NN; ++x) if (!v.used[x]) { v.used[x] = true;

Z[18] = Sum4-b-t-x;

if ((Z[18] < 0) || (Z[18] >= NN) || v.used[Z[18]]) {v.used[x] = false;

continue;}    v.used[Z[18]] = true;

for (int j = 0; j < NN; ++j) if (!v.used[j]) {  v.used[j] = true;

for (int o = 0; o < NN; ++o) if (!v.used[o]) {  v.used[o] = true;

Z[33]=Msum-j-b-o-a-t;

if ((Z[33] < 0) || (Z[33] >= NN) || v.used[Z[33]]) {v.used[o] = false; continue;}  v.used[Z[33]] = true;

for (int z = 0; z < NN; ++z) if (!v.used[z])

{ v.used[z] = true; Z[24]=Sum4-j-z-o;

if ((Z[24] < 0) || (Z[24] >= NN) || v.used[Z[24]]) {v.used[z] = false; continue;}  v.used[Z[24]] = true;

for (int w = 0; w < NN; ++w) if (!v.used[w]) {   v.used[w] = true;

Z[30]=b+j+o+t-w-Sum2;

if ((Z[30] < 0) || (Z[30] >= NN) || v.used[Z[30]]) {v.used[w] = false; continue;}     v.used[Z[30]] = true;

for (int p = 0; p < (NN-1); ++p) if (!v.used[p]) {   v.used[p] = true;

for (int q = p+1; q < NN; ++q) if (!v.used[q]) {   v.used[q] = true;

Z[5]=Sum4-o-p-q-j-t+w+e;
```

```
if ((Z[5] < 0) || (Z[5] >= NN) || v.used[Z[5]]) {v.used[q] = false; continue;}   v.used[Z[5]] = true; Z[35]=Sum2-b+p+q-x-e;

if ((Z[35] < 0) || (Z[35] >= NN) || v.used[Z[35]]) {      v.used[q] = false; v.used[Z[5]] = false; continue;}v.used[Z[35]] = true;

for (int h = 0; h < NN; ++h) if (!v.used[h]) {         v.used[h] = true; Z[28]=Sum4-p-q-h;

if ((Z[28] < 0) || (Z[28] >= NN) || v.used[Z[28]]) {v.used[h] = false; continue;}

v.used[Z[28]] = true;

for (int n = 0; n < NN; ++n) if (!v.used[n]) {           v.used[n] = true; Z[11]=Msum-j-z-n-q-h;

if ((Z[11] < 0) || (Z[11] >= NN) || v.used[Z[11]]) {v.used[n] = false; continue;}                  v.used[Z[11]] = true;

for (int d = 0; d <  NN; ++d) if (!v.used[d]) {     v.used[d] = true; Z[29]=Msum-Z[24]-p-d-o-Z[28];

if ((Z[29] < 0) || (Z[29] >= NN) || v.used[Z[29]]) {v.used[d] = false; continue;}

v.used[Z[29]] = true;

for (int g = 0; g < NN; ++g) if (!v.used[g]) {     v.used[g] = true;

Z[32]=a+b-d-g-n+Sum2;

if ((Z[32] < 0) || (Z[32] >= NN) || v.used[Z[32]]) {v.used[g] = false; continue;}

v.used[Z[32]] = true; Z[17]=d-a+g+n-p-q+x;

if ((Z[17] < 0) || (Z[17] >= NN) || v.used[Z[17]])

{v.used[g] = false; v.used[Z[32]] = false; continue;}

v.used[Z[17]] = true; Z[23]=a+b-g+j+o+p+q+t-w-Sum4;

if ((Z[23] < 0) || (Z[23] >= NN) || v.used[Z[23]])

{v.used[g] = false; v.used[Z[32]] = false;

v.used[Z[17]] = false; continue;}     v.used[Z[23]] = true;

for (int f = 0; f < NN; ++f) if (!v.used[f]) {     v.used[f] = true;

Z[4]=Msum-x-g-t-f-Z[5];

if ((Z[4] < 0) || (Z[4] >= NN) || v.used[Z[4]]) {v.used[f] = false; continue;}   v.used[Z[4]] = true;

for (int m = 0; m <NN; ++m) if (!v.used[m]) { v.used[m] = true;

Z[13]=Msum-m-w-e-a-Z[17];

if ((Z[13] < 0) || (Z[13] >= NN) || v.used[Z[13]]) {v.used[m] = false; continue;}
```

```
v.used[Z[13]] = true;

for (int k = 0; k < NN; ++k) if (!v.used[k]) {    v.used[k] = true;

Z[22]=Msum-k-b-Z[18]-Z[23]-Z[20];

if ((Z[22] >= 0) && (Z[22] < NN) && !v.used[Z[22]]) {   v.used[Z[22]] = true; Z[34]=Msum-m-q-Z[4]-Z[22]-Z[28];

if ((Z[34] >= 0) && (Z[34] < NN) && !v.used[Z[34]]) {   v.used[Z[34]] = true;

Z[31]=Msum-f-h-k-p-Z[13];

if ((Z[31] >= 0) && (Z[31] < NN) && !v.used[Z[31]]) {

Z[0]=x; Z[1]=f; Z[2]=g; Z[3]=t; Z[6]=z; Z[7]=h;

Z[8]=n; Z[9]=j; Z[10]=q; Z[12]=w; Z[14]=e; Z[15]=a;

Z[16]=m; Z[19]=k; Z[21]=b; Z[25]=p; Z[26]=d; Z[27]=o;

++count; writeSquare(Z, wfp);

if (++pcount == 1000000) {

printf("count %lu\n", count);

pcount = 0; fflush(wfp);}   }

v.used[Z[34]] = false;}      v.used[Z[22]] = false;}

v.used[k] = false;}    v.used[m] = false;    v.used[Z[13]] = false;}

v.used[f] = false;     v.used[Z[4]] = false;}

v.used[g] = false;v.used[Z[17]] = false;      v.used[Z[23]] = false; v.used[Z[32]] = false;}  v.used[d] = false; v.used[Z[29]]
= false;}

v.used[n] = false;v.used[Z[11]] = false;}   v.used[h] = false;

v.used[Z[28]] = false;}  v.used[q] = false; v.used[Z[5]] = false;

v.used[Z[35]] = false;} v.used[p] = false;}v.used[w] = false;

v.used[Z[30]] = false;}v.used[z] = false;  v.used[Z[24]] = false;}  v.used[o] = false; v.used[Z[33]] = false;}   v.used[j] =
false;}

v.used[x] = false;   v.used[Z[18]] = false;}   v.used[t] = false;}

printf("number of squares %d\n", count);

return count;}

 // makeSquares
```

```
voidget_rest_of_line(int c) {

if (c != '\n') do { c = getchar(); } while (c != '\n');}

voidgetNumPatterns(int *num) {

int unused = scanf("%d", num);

int c = getchar(); get_rest_of_line(c);}

boolcheck_abe(int a, int b, int e) {

boolrv = true;

if ((a <= 0) || (a > NN) ||  (b <= 0) || (b > NN) ||  (e <= 0) || (e > NN)) {

printf("\aValue range is 1 to %d.\n\n", NN);  rv = false;}

returnrv;}

/* check_abe */

boolcheckNum(intnum) {

boolrv = true;

if (num <= 0) {

printf("\aNumber must be a positive integer\n\n");

rv = false;  }

returnrv;}

constintbufSize = 128;

voidopenOutput(int a, int b, int e, char *wfpName, FILE **wfp) {

printf("************: abe: %d %d %d **************\n",a,b,e);

constintdefSize = 15;

charbuf[bufSize], buf1[bufSize], defaultName[defSize];

if (a == 0)

strcpy(defaultName, "s6_counts");

else

strcpy(defaultName, "s6_magic");

strcpy(buf, defaultName);  strcat(buf, ".txt");
```

```
 {int sub = 0;
/* Check if there already is a file of that name. */
do {
if ((fopen(buf, "r") == NULL) && (errno == ENOENT)) {
    /* No file. End the loop. */
break;   } else {
/* There is a file. Append "_1", "_2", ... until
the name does not match an existing file name. */
strcpy(buf, defaultName);
sprintf(buf1, "_%i", ++sub);
strcat(buf, buf1);  strcat(buf, ".txt");   }  }
while (true);
if ( (*wfp = fopen(buf, "w")) !=NULL) {
printf("\n%s file is %s\n", a == 0 ? "Data" : "Squares", buf);
strcpy(wfpName, buf);
 } else {
strcpy(buf1, "\a\nCan't open for write ");
strcat(buf1, buf);  perror(buf1);
 }}}
/* openOutput */
intgetSquares(int a0, int b0, int e0, int t0, int t1, intnum, FILE *wfpc)
{  charwfpsName[bufSize]; FILE *wfps = NULL;
intlinecount = 0; Uint count = 0; int patterns = 0;
{int a; for (a = --a0; a < NN; ++a) {
{int b; for (b = --b0; b < NN; ++b) if (a != b) {
{int e; for (e = --e0; e < NN; ++e) if ((a < e) && (e < b)) {
int J = Sum4-e-a-b;
```

Research Open Journal of Information Science and Application
Vol. 1, No. 1, August 2013, PP:01-19
Available online at http://www.scitecpub.com/Journals.php

```
if ((J <= a) || (J == b) || (J == e) || (J >= NN)) continue;

openOutput(a+1, b+1, e+1, wfpsName, &wfps);

if (wfps != NULL) { time_tstartTime; startTime =time(NULL);

count = makeSquares(a, b, e, t0, t1,  J, wfps);

fprintf(wfpc, "a: %2d b: %2d e: %2d number of squares: %10lu time: ",a+1, b+1, e+1, count);

{ intelapsed_t = (int)(time(NULL) - startTime);

inthr = elapsed_t/3600; elapsed_t %= 3600;

{ int min = elapsed_t/60, sec = elapsed_t%60;

{ char *fmt = "%3d:%02d:%02d\n";

printf(fmt, hr, min, sec); fprintf(wfpc, fmt, hr, min, sec);

if (++linecount == 5) { fprintf(wfpc, "\n"); linecount = 0; }

fflush(wfpc); fclose(wfps);

if (++patterns == num) return patterns;

}          }         }       }      }}    }}  }}  return patterns;

} /* getSquares */

int main(intargc, char** argv) {

if(argc< 5) {

printf("Usage %s <a><b><e><t0><t1>\n",argv[0]);

return 1;  }

int a=atoi(argv[1]) , b=atoi(argv[2]) , e=atoi(argv[3]),  t0=atoi(argv[4])   ,

t1=atoi(argv[5]);

printf("Input start values %d %d %d %d %d: \n",a,b,e,t0,t1);

if (check_abe(a, b, e)) {intnum = 1;char wfpcName[bufSize]; FILE *wfpc = NULL;

openOutput(0, 0, 0, wfpcName, &wfpc);

if (wfpc != NULL) {        time_tstartTime = time(NULL);

int patterns = getSquares(a, b, e, t0, t1, num, wfpc);

intelapsed_t = (int)(time(NULL) - startTime);
```

```
inthr = elapsed_t/3600; elapsed_t %= 3600;

{ int min = elapsed_t/60, sec = elapsed_t%60;

{ char *fmt = "\na, b, e patterns: %d elapsed time: %d:%02d:%02d t0:%02d t1:%02d\n";

printf(fmt, patterns, hr, min, sec, t0, t1);

fprintf(wfpc, fmt, patterns, hr, min, sec, t0, t1);

fclose(wfpc);

}   }  }  } return 0;}
```

Acknowledgment

Thanks are due to Riccardo Bruno for his major contributions by implementing the code in the EUMED GRID system.

References

[1] Al-Ashhab, S.: Magic Squares 5x5, the international journal of applied science and computations, Vol. 15, No.1, pages 53-64 (2008).

[2] Al-Ashhab, S.: Even-order Magic Squares with Special Properties, International Journal of Open Problems in Mathematics and Computer Science, Vol. 5, No. 2 (2012).

[3] Al-Ashhab, S.: Special Magic Squares of Order Six and Eight, International Journal of Digital Information and Wireless Communications (IJDIWC) 1(4): 769-781 (2012).

[4] Ahmed, M.: Algebraic Combinatorics of Magic Squares , Ph.D. Thesis, University Of California (2004).

[5] Ahmed, M.: How Many Squares Are There, Mr. Franklin?: Constructing and Enumerating Franklin Squares, American Mathematical Monthly 111, pages 394–410 (2004).

[6] Amela, M.: Structured 8 x 8 Franklin Squares, http://www.region.com.ar/amela/franklinsquares/

[7] Bellew, J.: Counting the Number of Compound and Nasik Magic Squares, Mathematics Today, pages 111-118 August (1997).

[8] Besslich, Ph. W.: Comments on Electronic Techniques for Pictorial Image Reproduction, IEEE Transactions on Communications 31, pages 846– 847 (1983).

[9] Besslich, Ph. W.: A Method for the Generation and processing of Dyadic Indexed Data, IEEE Transactions on Computers, C-32(5), pages 487– 494 (1983).

[10] Ollerenshaw, K., Brée, D. S., Most-perfect Pandiagonal Magic Squares: Their Construction and Enumeration, The Institute of Mathematics And its Applications, Southend-on-Sea, U.K., (1998).

[11] McClintock, E.: On the Most Perfect Forms of Magic Squares, with Methods for Their Production, American Journal of Mathematics 19, pages 99–120 (1897).

[12] Kolman, B.: Introductory Linear Algebra with Applications, 3rd edition (1991).

[13] Van den Essen, A.: Magic squares and linear algebra, American Mathematical Monthly 97, pp. 60-62 (1990).

[14] Walter Trump, www.trump.de/magic-squares