```
 1: program Sudoku_Statistics_2;
 2:
 3: // ...shows a 4*4 sudoku like calculation of the possibilities
 4: //solution is sqr (4!4!4!4!)  = 756 possibilities with diagonals = 48
 5: //shows the use of 2 dim arrays, loc's=105
 6:
 7: type
 8:     TPerm = array[1..4] of byte;
 9:     //TCube = array[1..6] of byte;
10:
11:
12: procedure sudok_permutation;
13: //solution is sqr 4!4!4!4!  = 756 with diagonal = 48
14: var
15:    perm: TPerm;
16:    resu: array[1..24] of TPerm;
17:    i,k,l,m: byte;
18:    count, suma: integer;
19:
20: begin
21: perm[1]:= 1;
22: perm[2]:= 2;
23: perm[3]:= 4;
24: perm[4]:= 8;
25: i:=1; k:=1;
26: l:=1; m:=1;
27: count:= 0;
28: suma:= 0;
29: writeln('((((((((((((((start of full sudokus))))))))))))))');
30:  for i:= 1 to 4 do
31:    for k:= 1 to 4 do
32:      for l:= 1 to 4 do
33:        for m:= 1 to 4 do begin
34:          //writeln(inttostr(perm[i])+inttostr(perm[k])+inttostr(perm[l])+
35:          //inttostr(perm[m]))
36:          //sure to make each number single
37:          suma:= perm[i]+perm[k]+perm[l]+perm[m]
38:          if suma = 15 then begin
39:          //writeln(inttoStr(suma))
40:            inc(count);
41:            //2 dim arrays
42:            resu[count][1]:= perm[i]
43:            resu[count][2]:= perm[k]
44:            resu[count][3]:= perm[l]
45:            resu[count][4]:= perm[m]
46:            writeln(inttostr(resu[count][1])+inttostr(resu[count][2])+
47:                    inttostr(resu[count][3])+inttostr(resu[count][4]))
48:          end;
49:             //write(inttostr(perm[m]));
50:           //write('-test-')
51:        end
52:        writeln('----------end of single lines-----------')
53:        writeln('');
54:
55:    count:= 0;
56:    suma:= 0;
57:    for i:= 1 to 24 do
58:      for k:= 1 to 24 do
59:        for l:= 1 to 24 do
60:          for m:= 1 to 24 do begin
61:            //writeln(inttostr(perm[i])+inttostr(perm[k])+inttostr(perm[l])+
62:              //inttostr(perm[m]))
63:             suma:=0;
64:             if (resu[i][1]+resu[k][1]+resu[l][1]+resu[m][1] = 15) and
65:                (resu[i][2]+resu[k][2]+resu[l][2]+resu[m][2] = 15) and
66:                (resu[i][3]+resu[k][3]+resu[l][3]+resu[m][3] = 15) and
67:                (resu[i][4]+resu[k][4]+resu[l][4]+resu[m][4] = 15) and
68:                //check also diagonals
69:                (resu[i][1]+resu[k][2]+resu[l][3]+resu[m][4] = 15) and
70:                (resu[m][1]+resu[l][2]+resu[k][3]+resu[i][4] = 15) then begin
71:              //writeln(inttoStr(suma))
72:                 inc(count);
73:                 write(inttostr(count))
74:                 //change the symbols here
75:                 writeln(inttostr(resu[i][1])+inttostr(resu[i][2])+
76:                         inttostr(resu[i][3])+inttostr(resu[i][4]))
```

```
77:                 writeln(inttostr(resu[k][1])+inttostr(resu[k][2])+
78:                         inttostr(resu[k][3])+inttostr(resu[k][4]))
79:                 writeln(inttostr(resu[l][1])+inttostr(resu[l][2])+
80:                         inttostr(resu[l][3])+inttostr(resu[l][4]))
81:                 writeln(inttostr(resu[m][1])+inttostr(resu[m][2])+
82:                         inttostr(resu[m][3])+inttostr(resu[m][4]))
83:
84:                 writeln('------(((((((((())))))))))------')
85:             end;
86:         end;
87:     writeln('4*4 sudokus: '+inttostr(count))
88: end;
89:
90:
91: // main sudo
92: begin
93: //sudo tester
94:   sudok_permutation;
95: end.
96:
97: just maXbox from
98:          ____    ___  _      ____    _   _   _
99:         |  _ \  |  _| | |    |  _ \  | | | | | |
100:        | | . | | |_  | |    | |_| | | |_| | | |
101:        | | | | |  _| | |    |  __/  |  _  | | |
102:        | |_. | | |_  | |__  | |     | | | | | |
103:        |____/  |___| |____| |_|     |_| |_| |_|
104:
105:
```