ALL SOLUTIONS TO THE CRAZY SHEEP GAME, by Neal R. Wagner

Pieces of Crazy Sheep Game (Sides are given counter-clockwise):

```
Piece number  0:      white rear,  black front, brown front, gray  rear
Piece number  1:      white rear,  brown front, gray  front, black rear
Piece number  2 = 4:  white rear,  brown rear,  black front, gray  front
Piece number  3:      white rear,  brown rear,  gray  front, black front
Piece number  4 = 2:  white rear,  brown rear,  black front, gray  front
Piece number  5:      white rear,  gray  rear,  black front, gray  front
Piece number  6:      white rear,  black rear,  gray  front, brown front
Piece number  7:      white rear,  gray  rear,  black front, brown front
Piece number  8:      white front, gray  rear,  brown rear,  black front
Piece number  9:      white front, brown rear,  black rear,  gray  front
Piece number 10 = 11: white front, brown front, black rear,  gray  rear
Piece number 11 = 10: white front, brown front, black rear,  gray  rear
Piece number 12:      white front, gray  front, black rear,  brown rear
Piece number 13:      white front, gray  front, brown rear,  black rear
Piece number 14:      white front, black front, black rear,  gray  rear
Piece number 15:      white front, brown front, black rear,  brown rear
```

In the table below, the first four numbers are the pieces in the corners, with the lowest numbered one first, and the rest given in counter-clockwise order. These were printed first to be sure there are no duplicates in the list of solutions. The next number, in parentheses, gives the orientation of the first piece listed, which is in the lower left corner (position 0). Codes for the orientation are: 0 = white down, 1 = white left, 2 = white up, 3 = white right.

Codes for the locations of the pieces:

```
       +----+----+----+----+
Row 3: | 12 | 13 | 14 | 15 |
       +----+----+----+----+
Row 2: |  8 |  9 | 10 | 11 |
       +----+----+----+----+
Row 1: |  4 |  5 |  6 |  7 |
       +----+----+----+----+
Row 0: |  0 |  1 |  2 |  3 |
       +----+----+----+----+
```

Table of all 55 solutions:

The sixteen final numbers for each row give the numbers of pieces in positions 0, 1, 2, 3, through 15, in the arrangement above. For each solution listed below, there are three other solutions that are rotations of the given solution by 90, 180, and 270. To get the solution, lay out the first piece in position 0 and in the proper orientation. Then each subsequent piece can be inserted, since there are no further choices.

(Because of a slight computer glitsch, I did some hand editing of the orientation data, so that it is possible that I got a few of the initial orientations reversed.)

It took about six hours to write and debug the program that found all solutions, though I was a bit lucky. Then it required only a fraction of a second for the machine to find the first solution. (The program used 314,152 position checks before reaching this first solution.) All 55 solutions were found in a few minutes.

Note that there is only one solution with piece number 8 in a corner, 2 solutions with piece 1 in a corner, and 2 solutions with piece 14 in a corner.

Corners	orie	Row 0	Row 1	Row 2	Row 3
0 3 15 12		0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15
0 9 2 6	(1)	0 2 14 9	12 8 3 10	15 7 10 5	6 13 1 2
0 9 3 5	(3)	0 12 7 9	1 10 2 8	13 2 15 14	5 6 10 3
0 9 3 5	(3)	0 15 2 9	1 10 7 8	13 2 12 14	5 6 10 3
0 9 3 5	(3)	0 15 7 9	1 10 2 8	13 2 10 14	5 6 12 3
0 9 3 6	(3)	0 10 5 9	1 12 2 8	13 7 15 14	6 2 10 3
0 9 3 6	(3)	0 12 5 9	1 10 2 8	13 2 15 14	6 7 10 3
0 9 3 6	(3)	0 12 7 9	1 10 2 8	13 5 15 14	6 2 10 3
0 9 3 6	(3)	0 15 2 9	1 10 7 8	13 5 12 14	6 2 10 3
0 9 3 6	(3)	0 15 5 9	1 10 2 8	13 2 10 14	6 7 12 3
0 9 3 6	(3)	0 15 7 9	1 10 2 8	13 5 10 14	6 2 12 3
0 9 3 7	(3)	0 12 5 9	1 10 2 8	13 2 15 14	7 6 10 3
0 9 3 7	(3)	0 15 5 9	1 10 2 8	13 2 10 14	7 6 12 3
0 9 12 3	(2)	0 5 1 9	10 13 2 14	8 7 10 6	3 15 2 12
0 9 13 10	(0)	0 12 2 9	15 5 14 1	2 10 6 7	10 3 8 13
0 9 13 10	(0)	0 15 2 9	12 2 14 1	5 10 6 7	10 3 8 13
0 9 13 15	(0)	0 10 2 9	12 2 14 1	7 10 6 5	15 3 8 13
0 9 15 3	(2)	0 5 1 9	10 13 2 14	8 7 10 6	3 12 2 15
1 12 2 9	(0)	1 13 2 12	7 10 5 14	3 8 10 6	9 0 15 2
1 14 5 9	(0)	1 13 2 14	10 7 10 6	3 8 15 2	9 0 12 5
2 3 9 7	(1)	2 1 13 3	10 5 10 6	8 14 2 15	7 12 0 9
2 5 12 6	(2)	2 0 13 5	14 15 1 10	8 3 7 2	6 9 10 12
2 5 12 9	(2)	2 0 13 5	14 15 1 10	8 3 7 2	9 6 10 12
2 6 2 10	(0)	2 10 9 6	12 7 3 8	5 1 15 14	10 13 0 2
2 6 14 8	(1)	2 1 13 6	7 10 5 15	12 2 10 3	8 0 9 14
2 8 5 9	(0)	2 10 1 8	12 7 13 14	2 15 6 3	9 0 10 5
2 9 2 10	(0)	2 10 6 9	12 7 3 8	5 1 15 14	10 13 0 2
2 9 6 10	(1)	2 12 2 9	14 5 10 1	13 7 15 0	10 3 8 6
2 9 7 12	(0)	2 15 0 9	10 5 1 8	2 10 13 14	12 6 3 7
2 9 12 13	(3)	2 15 0 9	6 10 8 3	14 5 10 7	13 1 2 12
2 10 6 5	(1)	2 15 7 10	10 3 8 12	9 14 2 0	5 1 13 6
2 10 12 15	(0)	2 7 9 10	10 1 14 3	5 13 2 8	15 6 0 12
2 10 15 12	(3)	2 10 7 10	6 13 1 5	0 2 14 9	12 8 3 15
2 13 6 9	(2)	2 0 1 13	14 12 5 10	8 7 10 3	9 2 15 6
2 13 6 9	(2)	2 0 1 13	14 15 5 10	8 2 10 3	9 7 12 6
2 13 6 9	(2)	2 0 1 13	14 15 7 10	8 2 12 3	9 5 10 6
2 15 6 5	(1)	2 10 7 15	10 3 8 12	9 14 2 0	5 1 13 6
2 15 10 12	(3)	2 10 5 15	6 13 1 7	0 2 14 9	12 8 3 10
3 5 10 9	(2)	3 1 12 5	14 10 0 15	8 2 6 2	9 7 13 10
3 5 10 9	(2)	3 6 2 5	14 13 10 15	8 1 7 2	9 0 12 10
3 5 10 9	(2)	3 6 12 5	14 13 7 15	8 1 10 2	9 0 2 10
3 5 10 9	(2)	3 10 1 5	14 12 0 15	8 7 6 2	9 2 13 10
3 5 10 9	(2)	3 10 1 5	14 15 0 12	8 2 6 2	9 7 13 10
3 5 10 9	(2)	3 12 1 5	14 10 0 15	8 2 6 2	9 7 13 10
3 7 10 9	(2)	3 1 12 7	14 10 0 15	8 2 6 2	9 5 13 10
3 7 10 9	(2)	3 10 1 7	14 15 0 12	8 2 6 2	9 5 13 10
3 7 10 9	(2)	3 12 1 7	14 10 0 15	8 2 6 2	9 5 13 10
3 10 5 9	(2)	3 1 12 10	14 10 0 2	8 2 6 15	9 7 13 5
3 10 5 9	(2)	3 6 2 10	14 13 10 2	8 1 7 15	9 0 12 5
3 10 5 9	(2)	3 6 12 10	14 13 7 2	8 1 10 15	9 0 2 5
3 10 5 9	(2)	3 10 1 10	14 12 0 2	8 7 6 15	9 2 13 5
3 10 5 9	(2)	3 10 1 10	14 15 0 2	8 2 6 12	9 7 13 5
3 10 5 9	(2)	3 12 1 10	14 10 0 2	8 2 6 15	9 7 13 5
3 10 7 9	(2)	3 1 12 10	14 10 0 2	8 2 6 15	9 5 13 7
3 10 7 9	(2)	3 10 1 10	14 15 0 2	8 2 6 12	9 5 13 7
3 10 7 9	(2)	3 12 1 10	14 10 0 2	8 2 6 15	9 5 13 7

Here is the computer program:

```
#include <stdio.h>
```

```
typedef enum {W, T, B, G} color;
typedef enum {F, R} direct;
typedef struct {
        color c[4];
        direct d[4];
} P_type;
P_type p[16]; /* pieces */

typedef struct {
        int piece; /* 0 to 15 */
        int orient; /* 0 to 3 */
} B_type;
B_type b[4][4];

int a[16]; /* remaining pieces */

void try (int, int[]);
int check(int, int, int);
void init_b(void);
void print_board(int);
void out24(int);

int count = 0;

void main()
{
        int i;
        init_b();
        for(i = 0; i < 16; i++) a[i] = i;
        try(16,a);
}

void try (int n, int a[16])
{
        int i;
        int at[16];
        int cp;
        int co;
        /* getchar();
        printf("Try. n:%2i,a:", n);
        for(i = 0; i < n; i++) printf("%2i,", a[i]);
        printf("\n"); */
        for(cp = 0; cp < n; cp++) { /* for each remaining piece */
                if (n > 1) {
                        for(i = 0; i < cp; i++) at[i] = a[i];
                        for(i = cp+1; i < n; i++) at[i-1] = a[i];
                }
                for(i = n; i < 16; i++) at[i] = 1000000;
                for(co = 0; co < 4; co++) { /* for each orientation */
                        if (check(n, a[cp], co)) {
                                /* count++;
                                if (count < 20) print_board(n);*/
                                /* if (n == 1) print_board(n);
                                else  */
                                if (n <= 1) {
                                        print_board(n);
                                }
                                if (n >= 1) try(n-1, at);
                        }
                }
        }
}

int check(int n, int ap, int co)
{
        int num = 16 - n;
        int i = (int) (num / 4);
        int j = num % 4;
        int apt, cot;
        /*      getchar();
                printf("Check. ap:%2i, co:%2i\n", ap, co); */
        b[i][j].piece = ap;
        b[i][j].orient = co;
        if (j > 0) {
                apt = b[i][j-1].piece;
                cot = b[i][j-1].orient;
                if (p[ap].c[(3+co)%4] != p[apt].c[(1+cot)%4]) return 0;
                if (p[ap].d[(3+co)%4] == p[apt].d[(1+cot)%4]) return 0;
        }
        if (i > 0) {
                apt = b[i-1][j].piece;
                cot = b[i-1][j].orient;
                if (p[ap].c[(0+co)%4] != p[apt].c[(2+cot)%4]) return 0;
                if (p[ap].d[(0+co)%4] == p[apt].d[(2+cot)%4]) return 0;
        }
        /* printf("  Check--success! n:%2i, ap:%2i, co:%2i\n", n, ap, co);
        print_board(n); */
        return 1;
}

void print_board(int n)
{
        int k, i, j;
        int c[4];
        int minc, minci;
        int num = 17 - n;
        c[0] = b[0][0].piece;
        c[1] = b[0][3].piece;
        c[2] = b[3][3].piece;
        c[3] = b[3][0].piece;
        for (i = 0; i < 4; i++)
                if (c[i] == 4) c[i] = 2;
        minc = c[0]; minci = 0;
        for (i = 1; i < 4; i++)
                if (c[i] < minc) {
                        minc = c[i];
                        minci = i;
                }
        if ((c[minci] == 2) && (c[(minci+2)%4] == 2))
                if ( c[(minci+1)%4] > c[(minci+3)%4] )
                        minci = (minci+2)%4;
        for (i = 0; i < 4; i++)
                printf("%2i ", c[(minci+i)%4]);
        printf(";");
        switch (minci) {
        case 0: printf("(%2i)", b[0][0].orient);
                for(i = 0; i < 4; i++) {
                        printf("| ");
                        for(j = 0; j < 4; j++)
                                out24(b[i][j].piece);
                }
                break;
        case 1: printf("(%2i)", (b[0][3].orient+3)%4);
                for(j = 3;j >= 0; j--) {
                        printf("| ");
                        for(i = 0; i < 4; i++)
                                out24(b[i][j].piece);
                }
                break;
        case 2: printf("(%2i)", (b[3][3].orient+2)%4);
                for(i = 3;i >= 0; i--) {
                        printf("| ");
```

```
                        for(j = 3;j >= 0; j--)
                                out24(b[i][j].piece);
                }
                break;
        case 3: printf("(%2i)", (b[3][0].orient+1)%4);
                for(j = 0; j < 4; j++) {
                        printf("| ");
                        for(i = 3;i >= 0; i--)
                                out24(b[i][j].piece);
                }
                break;
        }
        printf("|\n");
}

void out24(int n)
{
        if (n == 4) n = 2;
        printf("%2i ", n);
}

void init_b(void)
{
        int i;
        for(i = 0; i < 16; i++) p[i].c[0] = W;
        for(i = 0; i < 8; i++) {
                p[i].d[0] = R; p[i].d[2] = F;
        }
        for(i = 8; i < 16; i++) {
                p[i].d[0] = F; p[i].d[2] = R;
        }
        for(i = 0; i < 2; i++) {
                p[i].d[1] = F; p[i].d[3] = R;
        }
        for(i = 10; i < 16; i++) {
                p[i].d[1] = F; p[i].d[3] = R;
        }
        for(i = 2; i < 10; i++) {
                p[i].d[1] = R; p[i].d[3] = F;
        }
        p[ 0].c[1] = B; p[ 0].c[2] = T; p[ 0].c[3] = G;
        p[ 1].c[1] = T; p[ 1].c[2] = G; p[ 1].c[3] = B;
        p[ 2].c[1] = T; p[ 2].c[2] = B; p[ 2].c[3] = G;
        p[ 3].c[1] = T; p[ 3].c[2] = G; p[ 3].c[3] = B;
        p[ 4].c[1] = T; p[ 4].c[2] = B; p[ 4].c[3] = G;
        p[ 5].c[1] = G; p[ 5].c[2] = B; p[ 5].c[3] = G;
        p[ 6].c[1] = B; p[ 6].c[2] = G; p[ 6].c[3] = T;
        p[ 7].c[1] = G; p[ 7].c[2] = B; p[ 7].c[3] = T;
        p[ 8].c[1] = G; p[ 8].c[2] = T; p[ 8].c[3] = B;
        p[ 9].c[1] = T; p[ 9].c[2] = B; p[ 9].c[3] = G;
        p[10].c[1] = T; p[10].c[2] = B; p[10].c[3] = G;
        p[11].c[1] = T; p[11].c[2] = B; p[11].c[3] = G;
        p[12].c[1] = G; p[12].c[2] = B; p[12].c[3] = T;
        p[13].c[1] = G; p[13].c[2] = T; p[13].c[3] = B;
        p[14].c[1] = B; p[14].c[2] = B; p[14].c[3] = G;
        p[15].c[1] = T; p[15].c[2] = B; p[15].c[3] = T;
}
```