1. Intro. This program makes DLX data for MacMahon's problem of putting his 24 four-colored triangles into a hexagon, matching colors at the edges. The outer edge color is forced to be `a`. (It's a rewrite of the program that I wrote in September 2004.)

Actually I might as well make it more general, by allowing the hexagon to be replaced by any of the twelve double-size hexiamonds. The coordinates of the hexiamonds are specified on the command line.

I use the following coordinates for triangles: Those with apex at the top are (x,y); those with apex at the bottom are $(x,y)'$. If we think of a clock placed in the center of triangle (x,y), it has edge neighbors $(x,y)'$ at 2 o'clock, $(x,y-1)'$ at 6 o'clock, $(x-1,y)'$ at 10 o'clock; it sees its nearest upright neighbors $(x,y+1)$ at 1 o'clock, $(x+1,y)$ at 3 o'clock, $(x+1,y-1)$ at 5 o'clock, $(x,y-1)$ at 7 o'clock, $(x-1,y)$ at 9 o'clock, $(x-1,y+1)$ at 11 o'clock. The transformation $(x,y)\mapsto(-y,x+y)'$, $(x,y)'\mapsto(-y,x+y+1)$ rotates 60° about the lower left corner point of $(0,0)$. (Putting (x,y) and $(x,y)'$ together in a parallelogram, then slanting the parallelogram into a square, gives normal Cartesian coordinates for the squares.)

The hexagon consists of Δ triangles (x,y) for $0\le x,y\le 3$ and $2\le x+y\le 5$, together with the ∇ triangles $(x,y)'$ for $0\le x,y\le 3$ and $1\le x+y\le 4$. To specify it on the command line, say this:

```
macmahon-triangles-dlx 00+ 10 10+ 01 01+ 11
```

[It's inconvenient to use the character ‘′’ in a command line, so we use ‘+’.]

With change files I'll adapt the rules for edge matching. So I use a *mate* table that presently does nothing.

```
#include <stdio.h>
#include <stdlib.h>
  char piece[24][4];
  char occ[6][6], occp[6][6], edgeh[7][7], edgel[7][7], edger[7][7];
  char mate[256];
  main(int argc, char *argv[])
  {
    register int i, j, k, l, x, y, z;
    ⟨Set up the mate table 2⟩;
    ⟨Generate the piece table 3⟩;
    ⟨Process the command line 4⟩;
    ⟨Output the item-name line 7⟩;
    for (j = 0; j < 6; j++)
      for (k = 0; k < 6; k++) {
        if (occ[j][k]) ⟨Output the options for triangle (j, k) 8⟩;
        if (occp[j][k]) ⟨Output the options for triangle (j, k)′ 9⟩;
      }
    ⟨Output the options for the boundary 10⟩;
  }
```

2. ⟨Set up the *mate* table 2⟩ ≡

```
  mate['a'] = 'a';
  mate['b'] = 'b';
  mate['c'] = 'c';
  mate['d'] = 'd';
```

This code is used in section 1.

3. ⟨ Generate the *piece* table 3 ⟩ ≡

```
for (i = 0, j = 'a'; j ≤ 'd'; j++) {
  piece[i][0] = piece[i][1] = piece[i][2] = j, i++;
  for (k = 'a'; k ≤ 'd'; k++)
    if (j ≠ k) piece[i][0] = piece[i][1] = j, piece[i][2] = k, i++;
  for (k = j + 1; k ≤ 'd'; k++)
    for (l = k + 1; l ≤ 'd'; l++) {
      piece[i][0] = j, piece[i][1] = k, piece[i][2] = l, i++;
      piece[i][0] = j, piece[i][1] = l, piece[i][2] = k, i++;
    }
}
```

This code is used in section 1.

4. ⟨ Process the command line 4 ⟩ ≡

```
if (argc ≠ 7) {
  fprintf(stderr, "Usage:␣%s␣t1␣t2␣t3␣t3␣t4␣t5␣t6\n", argv[0]);
  exit(−1);
}
for (j = 1; j ≤ 6; j++) {
  x = 2 * (argv[j][0] − '0'), y = 2 * (argv[j][1] − '0');
  if (argv[j][2] ≡ '\0') z = 0;
  else if (argv[j][2] ≡ '+') z = 1;
  else {
    fprintf(stderr, "Triangle␣`%s'␣should␣have␣the␣form␣xy␣or␣xy+!\n", argv[j]);
    exit(−2);
  }
  if (x < 0 ∨ x > 4 ∨ y < 0 ∨ y > 4) {
    fprintf(stderr, "Triangle␣`%s'␣should␣have␣coordinates␣between␣0␣and␣3!\n", argv[j]);
    exit(−3);
  }
  ⟨ Set the occupied table from x and y 5 ⟩;
}
⟨ Set the edge tables from occ and occp 6 ⟩;
printf("|␣%s␣%s␣%s␣%s␣%s␣%s␣%s\n", argv[0], argv[1], argv[2], argv[3], argv[4], argv[5], argv[6]);
```

This code is used in section 1.

5. ⟨ Set the occupied table from x and y 5 ⟩ ≡

```
if (occ[x + z][y + z]) {
  fprintf(stderr, "Triangle␣`%s'␣has␣been␣specified␣twice!\n", argv[j]);
  exit(−4);
}
occ[x + z][y + z] = occp[x + z][y + z] = 1;
if (z) occp[x][y + 1] = occp[x + 1][y] = 1;
else occ[x][y + 1] = occ[x + 1][y] = 1;
```

This code is used in section 4.

6. ⟨ Set the edge tables from *occ* and *occp* 6 ⟩ ≡

```
for (x = 0; x < 6; x++)
  for (y = 0; y < 6; y++) {
    edgeh[x][y] += occ[x][y], edgel[x][y] += occ[x][y], edger[x][y] += occ[x][y];
    edgeh[x][y + 1] += occp[x][y], edgel[x][y] += occp[x][y], edger[x + 1][y] += occp[x][y];
  }
```

This code is used in section 4.

7. There's a primary item `*` for forcing the boundary condition. There's a primary item *xy* or *xy*`'` for each triangle. There's a primary item *abc* for each piece. There's a secondary item for each edge, denoting the color on that edge; the edges are `-`*xy*, `/`*xy*, `\`*xy* for the horizontal, forward-leaning, or backward-leaning edges that surround triangle (x, y).

⟨ Output the item-name line 7 ⟩ ≡

```
printf("*␣");
for (j = 0; j < 6; j++)
  for (k = 0; k < 6; k++) {
    if (occ[j][k]) printf("%d%d␣", j, k);
    if (occp[j][k]) printf("%d%d'␣", j, k);
  }
for (i = 0; i < 24; i++) printf("%s␣", piece[i]);
printf("|");
for (j = 0; j < 7; j++)
  for (k = 0; k < 7; k++) {
    if (edgeh[j][k]) printf("␣-%d%d", j, k);
    if (edger[j][k]) printf("␣/%d%d", j, k);
    if (edgel[j][k]) printf("␣\\%d%d", j, k);
  }
printf("\n");
```

This code is used in section 1.

8. ⟨ Output the options for triangle (j, k) 8 ⟩ ≡

```
for (i = 0; i < 24; i++) {
  printf("%d%d␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k, piece[i][0], j, k, piece[i][1], j, k,
      piece[i][2]);
  if (piece[i][1] ≠ piece[i][2]) {
    printf("%d%d␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k, piece[i][1], j, k, piece[i][2], j,
        k, piece[i][0]);
    printf("%d%d␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k, piece[i][2], j, k, piece[i][0], j,
        k, piece[i][1]);
  }
}
```

This code is used in section 1.

9. ⟨ Output the options for triangle $(j,k)'$ 9 ⟩ ≡

```
for (i = 0; i < 24; i++) {
  printf("%d%d'␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k + 1, mate[piece[i][0]], j + 1, k,
      mate[piece[i][1]], j, k, mate[piece[i][2]]);
  if (piece[i][1] ≠ piece[i][2]) {
    printf("%d%d'␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k + 1, mate[piece[i][1]], j + 1,
        k, mate[piece[i][2]], j, k, mate[piece[i][0]]);
    printf("%d%d'␣%s␣-%d%d:%c␣/%d%d:%c␣\\%d%d:%c\n", j, k, piece[i], j, k + 1, mate[piece[i][2]], j + 1,
        k, mate[piece[i][0]], j, k, mate[piece[i][1]]);
  }
}
```

This code is used in section 1.

10. The boundary edges all are colored `a`. (A text editor could change this.)

⟨ Output the options for the boundary 10 ⟩ ≡

```
printf("*");
for (j = 0; j < 7; j++)
  for (k = 0; k < 7; k++) {
    if (edgeh[j][k] ≡ 1) printf("␣-%d%d:%c", j, k, ¬occ[j][k] ? mate['a'] : 'a');
    if (edgel[j][k] ≡ 1) printf("␣\\%d%d:%c", j, k, ¬occ[j][k] ? mate['a'] : 'a');
    if (edger[j][k] ≡ 1) printf("␣/%d%d:%c", j, k, ¬occ[j][k] ? mate['a'] : 'a');
  }
printf("\n");
```

This code is used in section 1.

11. Index.

MACMAHON-TRIANGLES-DLX