Transformations

Russ Cox

We represent a board as a data structure containing the dimension and the contents. We pass around the data structure itself, not a reference to it, so that we can return new boards, and so on.

This makes it easy to define reflect and rotate operations that return reflected and rotated boards.

Once we have these, we just check to see what combination of transformations makes the old board into the new board.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#define MAXN 10

typedef struct Board Board;
struct Board {
    int n;
    char b[MAXN][MAXN];
};

/* rotate 90 degree clockwise: [r, c] -> [c, n - r] */
Board
rotate(Board b)
{
    Board nb;
    int r, c;

    nb.n = b.n;
    for(r=0; r<b.n; r++)
        for(c=0; c<b.n; c++)
            nb.b[c][b.n-1 - r] = b.b[r][c];
    return nb;
}

/* reflect board horizontally: [r, c] -> [r, n-1 -c] */
Board
reflect(Board b)
{
    Board nb;
    int r, c;

    nb.n = b.n;
    for(r=0; r<b.n; r++)
        for(c=0; c<b.n; c++)
            nb.b[r][b.n-1 - c] = b.b[r][c];
    return nb;
}

/* return non-zero if and only if boards are equal */
int
eqboard(Board b, Board bb)
{
    int r, c;
```

```
    if(b.n != bb.n)
        return 0;
    for(r=0; r<b.n; r++)
        for(c=0; c<b.n; c++)
            if(b.b[r][c] != bb.b[r][c])
                return 0;
    return 1;
}

Board
rdboard(FILE *fin, int n)
{
    Board b;
    int r, c;

    b.n = n;
    for(r=0; r<n; r++) {
        for(c=0; c<n; c++)
            b.b[r][c] = getc(fin);
        assert(getc(fin) == '\n');
    }
    return b;
}

void
main(void)
{
    FILE *fin, *fout;
    Board b, nb;
    int n, change;

    fin = fopen("transform.in", "r");
    fout = fopen("transform.out", "w");
    assert(fin != NULL && fout != NULL);

    fscanf(fin, "%d\n", &n);
    b = rdboard(fin, n);
    nb = rdboard(fin, n);

    if(eqboard(nb, rotate(b)))
        change = 1;
    else if(eqboard(nb, rotate(rotate(b))))
        change = 2;
    else if(eqboard(nb, rotate(rotate(rotate(b)))))
        change = 3;
    else if(eqboard(nb, reflect(b)))
        change = 4;
    else if(eqboard(nb, rotate(reflect(b)))
         || eqboard(nb, rotate(rotate(reflect(b))))
         || eqboard(nb, rotate(rotate(rotate(reflect(b))))))
        change = 5;
    else if(eqboard(nb, b))
        change = 6;
    else
        change = 7;

    fprintf(fout, "%d\n", change);

    exit(0);
}
```