§1 OTHELLO INTRO 1

1. Intro. I'm (hastily) writing this program in order to get some basic experience with a generalization of the game of Reversi (which is popularly called "Othello" in its principal variant).

I consider an $m \times n$ board, whose rows are numbered 1, 2, 3, ..., and whose columns are named a, b, c, A cell is identified by letter and digit. (I don't forbid m > 9 or n > 26; but bugs will show up if those parameters get too big.)

When prompted, the user should provide (on *stdin*) the name of the cell where the next move is to be made, followed by a newline (aka 'return'). The newline can optionally be preceded by '!', in which case the contents of the current board will be printed (on *stdout*).

The first four moves simply place pieces on the board; they are made without captures. (They are considered to be moves -3, -2, -1, and 0, so that move 1 will be the normal first move.) In standard Othello we have m = n = 9, and the preliminary moves are d5, e5, e4, d4; the first move might then be d3, in which case the color of d4 will be reversed.

This program actually allows more than two players. With p players, we start with move $1 - p^2$, and use the first p^2 moves to set up the initial position.

I don't attempt to do anything tricky. Cells of the board contain -1 if they're unoccupied, c if they are occupied and the top color is c. The colors are $0, 1, \ldots, p-1$. The value of board[i][j] is maintained for $0 \le i \le m+1$ and $0 \le j \le n+1$, but the boundary cells remain unoccupied.

```
#define m 8
                    /* this many rows */
#define n 8
                    /* this many columns */
                    /* this many players */
#define p = 2
#include <stdio.h>
#include <stdlib.h>
                                  /* cells of the current board position */
  char board[m+2][n+2];
                  /* the current move number */
  char deli[8] = \{-1, -1, -1, 0, 0, 1, 1, 1\}, delj[8] = \{-1, 0, 1, -1, 1, -1, 0, 1\};
                                                                                    /* the eight directions */
  char buffer[8]; /* used for input */
  int total[p];
                   /* how many pieces show this color? */
  \langle \text{Subroutines 3} \rangle;
  void main(void)
     register int i, j, k, l, pass, player;
     for (i = 0; i < m + 1; i ++)
       for (j = 0; j \le n + 1; j++) board [i][j] = -1;
     for (move = 1 - p * p, player = 0; ; move ++, player = (player + 1) \% p) {
       for (pass = 0; pass < p; pass \leftrightarrow) {
         for (i = 1; i \le m; i++)
            for (j = 1; j < n; j ++)
              if (islegal(i, j, player)) goto nextmove;
         printf("(player_{\square}\%c_{\square}cannot_{\square}move)\n", '0' + player));
         player = (player + 1) \% p;
                   /* the game is over: p passes in a row */
     nextmove: printf("Move_{\square}%d,_{\square}player_{\square}%c:_{\square}", move, `o' + player);
       fflush(stdout);
                           /* make sure the user sees the prompt */
       \langle Set i and j to the coordinates of the next move 2\rangle;
       makemove(i, j, player);
       if (buffer[2] \equiv "!") print\_board();
     print_board();
```

2 INTRO OTHELLO §2

```
\langle \text{Set } i \text{ and } j \text{ to the coordinates of the next move } 2 \rangle \equiv
  if (\neg fgets(buffer, 8, stdin)) {
     fprintf(stderr, "Unexpected\_end\_of\_input!\n");
     exit(-1);
  j = buffer[0] - 'a' + 1, i = buffer[1] - '0';
  if (i < 1 \lor i > m \lor j < 1 \lor j > n)
     fprintf(stderr, "Cell_{\square}'\%c\%c'_{\square}doesn't_{\square}exist!\n", buffer[0], buffer[1]);
     print_board();
     goto nextmove;
  if (\neg islegal(i, j, player)) {
     fprintf(stderr, "No!_{\sqcup}`\&c\&c`_{\sqcup}isn`t_{\sqcup}a_{\sqcup}legal_{\sqcup}move_{\sqcup}for_{\sqcup}\&c.\n", buffer[0], buffer[1], `o' + player);
     print_board();
     goto nextmove;
This code is used in section 1.
3. \langle \text{Subroutines } 3 \rangle \equiv
  void print_board(void)
     register i, j, k;
     for (k = 0; k < p; k++) total[k] = 0;
     for (i = 1; i \le m; i++) {
        for (j = 1; j \le n; j ++) {
          k = board[i][j];
          if (k \ge 0) total [k] ++;
          printf("%c", k < 0?', .': '0' + k);
        if (i \equiv m)
          for (k = 0; k < p; k \leftrightarrow) printf("\ldge", total[k]);
        printf("\n");
See also sections 4 and 6.
This code is used in section 1.
    This subroutine decides whether or not it's OK to place a piece of color c in cell (i, j) of the board. We
assume (without checking) that 1 \le i \le m, 1 \le j \le n, and 0 \le c < p.
\langle \text{Subroutines } 3 \rangle + \equiv
  int islegal(int i, int j, int c)
  {
     register int ii, jj, k, l;
     if (board[i][j] \ge 0) return 0;
                                              /* already occupied */
     if (move \le 0) return 1; /* we're just gettin' started */
     for (k = 0; k < 8; k++) (If direction k allows a move, return 1; otherwise continue 5);
     return 0:
  }
```

§5 OTHELLO INTRO 3

```
5. \langle \text{ If direction } k \text{ allows a move, } \mathbf{return } 1; \text{ otherwise } \mathbf{continue } 5 \rangle \equiv
  {
     for (ii = i + deli[k], jj = j + delj[k], l = 0; board[ii][jj] \ge 0; ii += deli[k], jj += delj[k], l++)
       if (board[ii][jj] \equiv c) goto maybe;
                       /* no occurrences of c in direction k */
     continue;
  maybe: if (l) return 1; /* yes, that move reverses at least l cells */
     continue;
                       /* two adjacent c's */
This code is used in section 4.
6. \langle \text{Subroutines } 3 \rangle + \equiv
  void makemove(int i, int j, int c)
     register int ii, jj, k;
     board[i][j] = c;
     if (move \le 0) return; /* just gettin' started */
     for (k = 0; k < 8; k++) (Do all reversals in direction k \neq 7);
7. \langle \text{ Do all reversals in direction } k \ 7 \rangle \equiv
     for (ii = i + deli[k], jj = j + delj[k]; board[ii][jj] \ge 0; ii += deli[k], jj += delj[k])
       if (board[ii][jj] \equiv c) goto reverse;
                      /* no occureeces of c in direction k */
  reverse: for (ii -= deli[k], jj -= delj[k]; ii \neq i \lor jj \neq j; ii -= deli[k], jj -= delj[k]) board [ii][jj] = c;
  }
This code is used in section 6.
```

4 INDEX OTHELLO §8

8. Index.

```
board: 1, 3, 4, 5, 6, 7.
buffer: \underline{1}, \underline{2}.
c: \underline{4}, \underline{6}.
deli: 1, 5, 7.
delj: \underline{1}, 5, 7.
exit: 2.
fflush: 1.
fgets: 2.
fprintf: 2.
i: \ \underline{1}, \ \underline{3}, \ \underline{4}, \ \underline{6}.
ii: \underline{4}, 5, \underline{6}, 7.
is legal: 1, 2, \underline{4}.
j: \ \underline{1}, \ \underline{3}, \ \underline{4}, \ \underline{6}.
jj: \underline{4}, \underline{5}, \underline{6}, \underline{7}.
m: \underline{1}.
main: \underline{1}.
makemove: 1, \underline{6}.
maybe: \underline{5}.
move: \underline{1}, 4, 6.
n: \underline{1}.
nextmove: \underline{1}, \underline{2}.
p: <u>1</u>.
pass: \underline{1}.
player: \underline{1}, \underline{2}.
print\_board\colon \ 1,\ 2,\ \underline{3}.
printf: 1, 3.
reverse: 7.
stderr: 2.
stdin: 1, 2.
stdout: 1.
total: \underline{1}, \underline{3}.
```

OTHELLO NAMES OF THE SECTIONS 5

```
\langle Do all reversals in direction k 7\rangle Used in section 6. \langle If direction k allows a move, return 1; otherwise continue 5\rangle Used in section 4. \langle Set i and j to the coordinates of the next move 2\rangle Used in section 1. \langle Subroutines 3, 4, 6\rangle Used in section 1.
```

OTHELLO

	Section	Pag
Intro		
Index	8	