

(See <https://cs.stanford.edu/~knuth/programs.html> for date.)

**1. Data for dancing.** This program creates data suitable for the DANCE routine, given the description of a board to be covered and a set of polyomino shapes.

The first line of input names all the board positions, in any order. Each position is a two-digit number representing  $x$  and  $y$  coordinates; each “digit” is a single character, 0–9 or a–z representing the numbers 0–35. For example,

11 12 13 21 22 23 31 32 33

is one way to describe a  $3 \times 3$  board.

The second line of input names all the pieces. Each piece name consists of at most three characters; the name should also be distinguishable from a board position. (The program does not check this.)

The remaining lines of input describe the polyominoes. First comes the name, followed by two integers  $s$  and  $t$ , meaning that the shape should appear in  $s$  rotations and  $t$  transpositions. Then come two-digit coordinates for each cell of the shape. For example, the line

P 4 2 00 10 01 11 02

describes a pentomino that can appear in 8 orientations; it is equivalent to eight lines

P 1 1 00 10 01 11 02  
P 1 1 00 10 01 11 21  
P 1 1 10 01 11 02 12  
P 1 1 00 10 20 11 21  
P 1 1 00 01 10 11 20  
P 1 1 00 01 02 11 12  
P 1 1 01 10 11 20 21  
P 1 1 00 01 10 11 12

obtained by rotating the original shape, then transposing and rotating again. The values of  $s$  and  $t$  depend on the symmetry of the piece; six cases (1, 1), (1, 2), (2, 1), (2, 2), (4, 1), and (4, 2) can arise, for pieces with full symmetry, swastika symmetry, double-reflection symmetry,  $180^\circ$  symmetry, reflection symmetry, and no symmetry. If  $s$  had been 2 instead of 4, only the first, second, fifth, and sixth of these eight orientations would have been generated.

After optional rotation and/or translation, each piece is translated in all possible ways that fit on the given board, by adding constant values  $(x, y)$  to all of its coordinate pairs. For example, if the piece P 1 1 00 10 01 11 02 is specified with the  $3 \times 3$  board considered above, it will lead to two possible rows in the exact cover problem, namely P 11 21 12 22 13 and P 21 31 22 23.

```
#define max_pieces 100    /* at most this many shapes */
#define buf_size 36*36*3+8 /* upper bound on line length */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
  <Global variables 4>
  <Subroutines 3>;
main()
{
  register char *p, *q;
  register int j, k, n, x, y, z;
  <Read and output the board 2>;
  <Read and output the piece names 5>;
  <Read and output the pieces 6>;
}
```

```

2. #define panic(m)
    { fprintf(stderr, "%s!\n%s", m, buf); exit(-1); }

⟨Read and output the board 2⟩ ≡
    fgets(buf, buf_size, stdin);
    if (buf[strlen(buf) - 1] ≠ '\n') panic("Input_line_too_long");
    bxmin = bymin = 35; bxmax = bymax = 0;
    for (p = buf; *p; p += 3) {
        while (isspace(*p)) p++;
        if (¬*p) break;
        x = decode(*p);
        if (x < 0) panic("Bad_x_coordinate");
        y = decode*(p + 1);
        if (y < 0) panic("Bad_y_coordinate");
        if (¬isspace*(p + 2)) panic("Bad_board_position");
        if (board[x][y]) panic("Duplicate_board_position");
        if (x < bxmin) bxmin = x;
        if (x > bxmax) bxmax = x;
        if (y < bymin) bymin = y;
        if (y > bymax) bymax = y;
        board[x][y] = 1;
    }
    if (bxmin > bxmax) panic("Empty_board");
    fwrite(buf, 1, strlen(buf) - 1, stdout);    /* output all but the newline */

```

This code is used in section 1.

```

3. ⟨Subroutines 3⟩ ≡
    int decode(c)
        char c;
    {
        if (c ≤ '9') {
            if (c ≥ '0') return c - '0';
        } else if (c ≥ 'a') {
            if (c ≤ 'z') return c + 10 - 'a';
        }
        return -1;
    }

```

See also section 12.

This code is used in section 1.

```

4. ⟨Global variables 4⟩ ≡
    char buf[buf_size];
    int board[36][36];    /* cells present */
    int bxmin, bxmax, bymin, bymax;    /* used portion of the board */

```

See also section 7.

This code is used in section 1.

```

5. ⟨Read and output the piece names 5⟩ ≡
    if (¬fgets(buf, buf_size, stdin)) panic("No_piece_names");
    printf("_ %s", buf);    /* just pass the piece names through */

```

This code is used in section 1.

6.  $\langle$  Read and output the pieces 6  $\rangle \equiv$

```

while (fgets(buf, buf_size, stdin)) {
  if (buf[strlen(buf) - 1] != '\n') panic("Input_line_too_long");
  for (p = buf; isspace(*p); p++) ;
  if (!*p) panic("Empty_line");
  for (q = p + 1; !isspace(*q); q++) ;
  if (q > p + 3) panic("Piece_name_too_long");
  for (q = name; !isspace(*p); p++, q++) *q = *p;
  *q = '\0';
  for (p++; isspace(*p); p++) ;
  s = *p - '0';
  if ((s != 1 & s != 2 & s != 4) || !isspace(*(p + 1))) panic("Bad_s_value");
  for (p += 2; isspace(*p); p++) ;
  t = *p - '0';
  if ((t != 1 & t != 2) || !isspace(*(p + 1))) panic("Bad_t_value");
  n = 0;
  xmin = ymin = 35; xmax = ymax = 0;
  for (p += 2; *p; p += 3, n++) {
    while (isspace(*p)) p++;
    if (!*p) break;
    x = decode(*p);
    if (x < 0) panic("Bad_x_coordinate");
    y = decode(*(p + 1));
    if (y < 0) panic("Bad_y_coordinate");
    if (!isspace(*(p + 2))) panic("Bad_board_position");
    if (n == 36 * 36) panic("Pigeonhole_principle_says_you_repeated_a_position");
    xx[n] = x, yy[n] = y;
    if (x < xmin) xmin = x;
    if (x > xmax) xmax = x;
    if (y < ymin) ymin = y;
    if (y > ymax) ymax = y;
  }
  if (n == 0) panic("Empty_piece");
   $\langle$  Generate the possible piece placements 8  $\rangle$ ;
}

```

This code is used in section 1.

7.  $\langle$  Global variables 4  $\rangle + \equiv$

```

char name[4]; /* name of current piece */
int s, t; /* symmetry type of current piece */
int xx[36 * 36], yy[36 * 36]; /* coordinates of current piece */
int xmin, xmax, ymin, ymax; /* range of coordinates */

```

8.  $\langle$  Generate the possible piece placements 8  $\rangle \equiv$   
**while** ( $t$ ) {  
     **for** ( $k = 1$ ;  $k \leq 4$ ;  $k++$ ) {  
         **if** ( $k \leq s$ )  $\langle$  Output translates of the current piece 11  $\rangle$ ;  
          $\langle$  Rotate the current piece 10  $\rangle$ ;  
     }  
      $\langle$  Transpose the current piece 9  $\rangle$ ;  
      $t--$ ;  
 }

This code is used in section 6.

9.  $\langle$  Transpose the current piece 9  $\rangle \equiv$   
**for** ( $j = 0$ ;  $j < n$ ;  $j++$ ) {  
      $z = xx[j]$ ;  
      $xx[j] = yy[j]$ ;  
      $yy[j] = z$ ;  
 }  
 $z = xmin$ ;  $xmin = ymin$ ;  $ymin = z$ ;  
 $z = xmax$ ;  $xmax = ymax$ ;  $ymax = z$ ;

This code is used in section 8.

10.  $\langle$  Rotate the current piece 10  $\rangle \equiv$   
**for** ( $j = 0$ ;  $j < n$ ;  $j++$ ) {  
      $z = xx[j]$ ;  
      $xx[j] = 35 - yy[j]$ ;  
      $yy[j] = z$ ;  
 }  
 $z = xmin$ ;  $xmin = 35 - ymax$ ;  $ymax = xmax$ ;  $xmax = 35 - ymin$ ;  $ymin = z$ ;

This code is used in section 8.

11.  $\langle$  Output translates of the current piece 11  $\rangle \equiv$   
**for** ( $x = bxmin - xmin$ ;  $x \leq bxmax - xmax$ ;  $x++$ )  
     **for** ( $y = bymin - ymin$ ;  $y \leq bymax - ymax$ ;  $y++$ ) {  
         **for** ( $j = 0$ ;  $j < n$ ;  $j++$ )  
             **if** ( $\neg board[x + xx[j]][y + yy[j]]$ ) **goto** *nope*;  
          $printf(name)$ ;  
         **for** ( $j = 0$ ;  $j < n$ ;  $j++$ )  $printf("\%c\%c", encode(x + xx[j]), encode(y + yy[j]))$ ;  
          $printf("\n")$ ;  
     *nope*: ;  
 }

This code is used in section 8.

12.  $\langle$  Subroutines 3  $\rangle + \equiv$   
**char** *encode*( $x$ )  
     **int**  $x$ ;  
 {  
     **if** ( $x < 10$ ) **return** '0' +  $x$ ;  
     **return** 'a' - 10 +  $x$ ;  
 }

**13. Index.**

*board*: 2, 4, 11.  
*buf*: 2, 4, 5, 6.  
*buf\_size*: 1, 2, 4, 5, 6.  
*bxmax*: 2, 4, 11.  
*bxmin*: 2, 4, 11.  
*bymax*: 2, 4, 11.  
*bymin*: 2, 4, 11.  
*c*: 3.  
*decode*: 2, 3, 6.  
*encode*: 11, 12.  
*exit*: 2.  
*fgets*: 2, 5, 6.  
*fprintf*: 2.  
*fwrite*: 2.  
*isspace*: 2, 6.  
*j*: 1.  
*k*: 1.  
*main*: 1.  
*max\_pieces*: 1.  
*n*: 1.  
*name*: 6, 7, 11.  
*nope*: 11.  
*p*: 1.  
*panic*: 2, 5, 6.  
*printf*: 5, 11.  
*q*: 1.  
*s*: 7.  
*stderr*: 2.  
*stdin*: 2, 5, 6.  
*stdout*: 2.  
*strlen*: 2, 6.  
*t*: 7.  
*x*: 1, 12.  
*xmax*: 6, 7, 9, 10, 11.  
*xmin*: 6, 7, 9, 10, 11.  
*xx*: 6, 7, 9, 10, 11.  
*y*: 1.  
*ymax*: 6, 7, 9, 10, 11.  
*ymin*: 6, 7, 9, 10, 11.  
*yy*: 6, 7, 9, 10, 11.  
*z*: 1.

- ⟨ Generate the possible piece placements 8 ⟩ Used in section 6.
- ⟨ Global variables 4, 7 ⟩ Used in section 1.
- ⟨ Output translates of the current piece 11 ⟩ Used in section 8.
- ⟨ Read and output the board 2 ⟩ Used in section 1.
- ⟨ Read and output the piece names 5 ⟩ Used in section 1.
- ⟨ Read and output the pieces 6 ⟩ Used in section 1.
- ⟨ Rotate the current piece 10 ⟩ Used in section 8.
- ⟨ Subroutines 3, 12 ⟩ Used in section 1.
- ⟨ Transpose the current piece 9 ⟩ Used in section 8.

# POLYOMINOES

	Section	Page
Data for dancing .....	<a href="#">1</a>	1
Index .....	<a href="#">13</a>	5