

1. Intro. This program makes DLX3 data for an interesting problem posed by Ian Cullis in 2022: Fill a 10×10 array with 1s, 2s, 3s, 4s so that there are exactly k occurrences of k in each row and each column. Also the 2s should form nontouching dominoes, the 3s should form nontouching trominoes, and the 4s should form nontouching ell-tetrominoes, where “nontouching” means not having edges in common.

This program is to be used with the UNIX command line

```
cat ian.dat | polyomino-dlx | ian-dlx
```

so that *stdin* contains appropriate data about the possible configurations of individual polyominoes and their boundaries.

```
#define bufsize 1024
#include <stdio.h>
#include <stdlib.h>
char buf[bufsize];
main()
{
    register int i, j, k;
    <Print the item-name line 2>;
    <Print the options for individual cells 3>;
    <Print the options for vetting polyominoes 4>;
}
```

2. There are primary items R_{ik} and C_{jk} for $0 \leq i, j < 10$ and $1 \leq k \leq 4$, indicating the number of k s in row or column k . There also are primary items $\#_{ij}$, meaning that cell ij has been “vetted” as a polyomino that matches its number.

There are secondary items ijk , which are essentially Boolean variables that state whether or not cell ij contains k .

I’ve also added primary items ij , with four options apiece. These aren’t necessary, but they speed up the search.

```
<Print the item-name line 2> ≡
for (i = 0; i < 10; i++)
    for (j = 0; j < 10; j++) printf("%d%d_\n", i, j);
for (i = 0; i < 10; i++)
    for (k = 1; k ≤ 4; k++) printf("%d|R%d%d_\n", k, i, k, k, i, k);
for (i = 0; i < 10; i++)
    for (j = 0; j < 10; j++) printf("#%d%d_\n", i, j);
printf("\n");
for (i = 0; i < 10; i++)
    for (j = 0; j < 10; j++)
        for (k = 1; k ≤ 4; k++) printf("_%d%d%d\n", i, j, k);
printf("\n");
```

This code is used in section 1.

3. \langle Print the options for individual cells 3 $\rangle \equiv$

```

for ( $i = 0$ ;  $i < 10$ ;  $i++$ )
  for ( $j = 0$ ;  $j < 10$ ;  $j++$ ) {
    printf ("%d%d_R%d1_C%d1_%d%d1:1_%d%d2:0_%d%d3:0_%d%d4:0\n",  $i, j, i, j, i, j, i, j, i, j$ );
    printf ("%d%d_R%d2_C%d2_%d%d1:0_%d%d2:1_%d%d3:0_%d%d4:0\n",  $i, j, i, j, i, j, i, j, i, j$ );
    printf ("%d%d_R%d3_C%d3_%d%d1:0_%d%d2:0_%d%d3:1_%d%d4:0\n",  $i, j, i, j, i, j, i, j, i, j$ );
    printf ("%d%d_R%d4_C%d4_%d%d1:0_%d%d2:0_%d%d3:0_%d%d4:1\n",  $i, j, i, j, i, j, i, j, i, j$ );
  }

```

This code is used in section 1.

```

4. #define less_one(k) (buf[k] == 'a' ? 9 : buf[k] - '1')
(Print the options for vetting polyominoes 4) ==
while (1) {
    if (!fgets(buf, bufsize, stdin)) break;
    switch (buf[0]) {
        case '|': case '_': continue;
        case 'o': i = less_one(2), j = less_one(3);
            printf("#%d%d%d1:1", i, j, i, j);
            break;
        case 'd':
            for (k = 1; buf[k] == '_'; k += 3) {
                i = less_one(k + 1), j = less_one(k + 2);
                if (buf[k + 3] == 'b') {
                    k++;
                    if (i ≥ 0 ∧ i < 10 ∧ j ≥ 0 ∧ j < 10) printf("%d%d2:0_", i, j);
                } else {
                    printf("#%d%d%d2:1_", i, j, i, j);
                }
            }
            break;
        case 'v': case 't':
            for (k = 1; buf[k] == '_'; k += 3) {
                i = less_one(k + 1), j = less_one(k + 2);
                if (buf[k + 3] == 'b') {
                    k++;
                    if (i ≥ 0 ∧ i < 10 ∧ j ≥ 0 ∧ j < 10) printf("%d%d3:0_", i, j);
                } else {
                    printf("#%d%d%d3:1_", i, j, i, j);
                }
            }
            break;
        case 'l':
            for (k = 1; buf[k] == '_'; k += 3) {
                i = less_one(k + 1), j = less_one(k + 2);
                if (buf[k + 3] == 'b') {
                    k++;
                    if (i ≥ 0 ∧ i < 10 ∧ j ≥ 0 ∧ j < 10) printf("%d%d4:0_", i, j);
                } else {
                    printf("#%d%d%d4:1_", i, j, i, j);
                }
            }
            break;
        default: fprintf(stderr, "Bad_input_line!_%s", buf);
    }
    printf("\n");
}

```

This code is used in section 1.

5. Index.

buf: [1](#), [4](#).

bufsize: [1](#), [4](#).

fgets: [4](#).

fprintf: [4](#).

i: [1](#).

j: [1](#).

k: [1](#).

less_one: [4](#).

main: [1](#).

printf: [2](#), [3](#), [4](#).

stderr: [4](#).

stdin: [1](#), [4](#).

- ⟨ Print the item-name line [2](#) ⟩ Used in section [1](#).
- ⟨ Print the options for individual cells [3](#) ⟩ Used in section [1](#).
- ⟨ Print the options for vetting polyominoes [4](#) ⟩ Used in section [1](#).

IAN-DLX

	Section	Page
Intro	1	1
Index	5	4