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

1. Data for dancing. This program creates data suitable for the DANCE routine, solving the famous "*n* queens problem." The value of *n* is a command-line parameter.

```
#include <stdio.h>
#include <stdlib.h>
  ⟨Global variables 3⟩
  ⟨Subroutines 5⟩;
  main(argc, argv)
      int argc;
      char ∗argv[];
  {
    register int j, k, n, nn, t;
    ⟨Read the command line 2⟩;
    ⟨Output the column names 4⟩;
    ⟨Output the possible queen moves 6⟩;
  }
```

2. ⟨Read the command line 2⟩ ≡

```
  if (argc ≠ 2 ∨ sscanf(argv[1], "%d", &param) ≠ 1) {
    fprintf(stderr, "Usage:␣%s␣n\n", argv[0]);
    exit(−1);
  }
  n = param;
  nn = n + n − 2;
```

This code is used in section 1.

3. ⟨Global variables 3⟩ ≡

```
  int param;
```

This code is used in section 1.

4. We process the cells of the board in "organ pipe order," on the assumption that—all other things being equal—a move near the center yields more constraints on the subsequent search.

```
⟨Output the column names 4⟩ ≡
  for (j = 0; j < n; j++) {
    t = (j & 1 ? n − 1 − j : n + j) ≫ 1;
    printf("r%c␣c%c␣", encode(t), encode(t));
  }
  printf("|");
  for (j = 1; j < nn; j++) printf("␣a%c␣b%c", encode(j), encode(j));
  printf("\n");
```

This code is used in section 1.

5. ⟨Subroutines 5⟩ ≡

```
  char encode(x)
      int x;
  {
    if (x < 10) return '0' + x;
    return 'a' − 10 + x;
  }
```

This code is used in section 1.

6. ⟨ Output the possible queen moves 6 ⟩ ≡

```
for (j = 0; j < n; j++)
  for (k = 0; k < n; k++) {
    printf("r%c␣c%c", encode(j), encode(k));
    t = j + k;
    if (t ∧ (t < nn)) printf("␣a%c", encode(t));
    t = n − 1 − j + k;
    if (t ∧ (t < nn)) printf("␣b%c", encode(t));
    printf("\n");
  }
```

This code is used in section 1.

7. Index.

QUEENS