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

1. Intro. This program produces a DLX file that corresponds to the problem of filling an $m \times n$ array with words, given a list of m-letter words for the columns and a list of n-letter words for the rows. It's supposed to compete with the program BACK-MXN-WORDS-NEW, which uses trie structures instead of dancing links.

As in that program, we specify the two lists of words on the command line, by naming the files that contain them. Those word lists can be limited by appending, say, '`:500`' to the file name if we want to use only 500 of its words.

```
#define infinity 100000000
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
  int maxmm = infinity, maxnn = infinity;
  FILE *mfile, *nfile;
  char mbuf[80], nbuf[80];
  main(int argc, char *argv[])
  {
    register int i, j, k, m, n;
    register char *w;
    ⟨ Process the command line 2 ⟩;
    ⟨ Output the DLX column-name line 3 ⟩;
    ⟨ Input the m-words and output the corresponding rows 4 ⟩;
    ⟨ Input the n-words and output the corresponding rows 5 ⟩;
  }
```

2. ⟨Process the command line 2⟩ ≡

```
if (argc ≠ 3) {
  fprintf(stderr, "Usage:␣%s␣mwords[:mm]␣nwords[:nn]\n", argv[0]);
  exit(−1);
}
w = strchr(argv[1], ':');
if (w) {      /* colon in filename */
  if (sscanf(w + 1, "%d", &maxmm) ≠ 1) {
    fprintf(stderr, "I␣can't␣parse␣the␣m-file␣spec␣`%s'!\n", argv[1]);
    exit(−20);
  }
  *w = 0;
}
if (¬(mfile = fopen(argv[1], "r"))) {
  fprintf(stderr, "I␣can't␣open␣file␣`%s'␣for␣reading␣m-words!\n", argv[1]);
  exit(−2);
}
if (¬fgets(mbuf, 80, mfile)) {
  fprintf(stderr, "The␣file␣`%s'␣of␣m-words␣is␣empty!\n", argv[1]);
  exit(−222);
}
m = strchr(mbuf, '\n') − mbuf;
if (m > 16) {
  fprintf(stderr, "I␣don't␣allow␣m=%d,␣since␣that's␣bigger␣than␣16!\n", m);
  exit(−666);
}
w = strchr(argv[2], ':');
if (w) {      /* colon in filename */
  if (sscanf(w + 1, "%d", &maxnn) ≠ 1) {
    fprintf(stderr, "I␣can't␣parse␣the␣n-file␣spec␣`%s'!\n", argv[1]);
    exit(−22);
  }
  *w = 0;
}
if (¬(nfile = fopen(argv[2], "r"))) {
  fprintf(stderr, "I␣can't␣open␣file␣`%s'␣for␣reading␣n-words!\n", argv[2]);
  exit(−3);
}
if (¬fgets(nbuf, 80, nfile)) {
  fprintf(stderr, "The␣file␣`%s'␣of␣n-words␣is␣empty!\n", argv[1]);
  exit(−333);
}
n = strchr(nbuf, '\n') − nbuf;
if (n > 16) {
  fprintf(stderr, "I␣don't␣allow␣n=%d,␣since␣that's␣bigger␣than␣16!\n", n);
  exit(−666);
}
printf("|␣word-rect-dlx");
if (maxmm < infinity) printf("␣%s:%d", argv[1], maxmm);
else printf("␣%s", argv[1]);
if (maxnn < infinity) printf("␣%s:%d\n", argv[2], maxnn);
else printf("␣%s\n", argv[2]);
```

This code is used in section 1.

3. This exact cover problem has $m+n$ primary columns Ri and Cj, and mn secondary columns ij, where i and j are encoded in hexadecimal notation; $0 \le i < m$ and $0 \le j < n$.

⟨ Output the DLX column-name line 3 ⟩ ≡

```
for (i = 0; i < m; i++) printf("␣R%x", i);
for (j = 0; j < n; j++) printf("␣C%x", j);
printf("␣|");
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++) printf("␣%x%x", i, j);
printf("\n");
```

This code is used in section 1.

4. ⟨ Input the m-words and output the corresponding rows 4 ⟩ ≡

```
for (k = 1; ; k++) {
  for (j = 0; j < n; j++) {
    printf("C%x", j);
    for (i = 0; i < m; i++) printf("␣%x%x:%c", i, j, mbuf[i]);
    printf("\n");
  }
  if (¬fgets(mbuf, 80, mfile)) break;
  if (k ≡ maxmm) break;
}
fprintf(stderr, "OK,␣I've␣read␣%d␣words␣from␣file␣`%s',\n", k, argv[1]);
```

This code is used in section 1.

5. ⟨ Input the n-words and output the corresponding rows 5 ⟩ ≡

```
for (k = 1; ; k++) {
  for (i = 0; i < m; i++) {
    printf("R%x", i);
    for (j = 0; j < n; j++) printf("␣%x%x:%c", i, j, nbuf[j]);
    printf("\n");
  }
  if (¬fgets(nbuf, 80, nfile)) break;
  if (k ≡ maxnn) break;
}
fprintf(stderr, "␣and␣read␣%d␣words␣from␣file␣`%s'.\n", k, argv[2]);
```

This code is used in section 1.

6. Index.

argc: 1, 2.
argv: 1, 2, 4, 5.
exit: 2.
fgets: 2, 4, 5.
fopen: 2.
fprintf: 2, 4, 5.
i: 1.
infinity: 1, 2.
j: 1.
k: 1.
m: 1.
main: 1.
maxmm: 1, 2, 4.
maxnn: 1, 2, 5.
mbuf: 1, 2, 4.
mfile: 1, 2, 4.
n: 1.
nbuf: 1, 2, 5.
nfile: 1, 2, 5.
printf: 2, 3, 4, 5.
sscanf: 2.
stderr: 2, 4, 5.
strchr: 2.
w: 1.

⟨ Input the m-words and output the corresponding rows 4 ⟩ Used in section 1.
⟨ Input the n-words and output the corresponding rows 5 ⟩ Used in section 1.
⟨ Output the DLX column-name line 3 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.

WORD-RECT-DLX

	Section	Page
Intro	1	1
Index	6	4