1. Intro. Given the specification of a sudoku puzzle in *stdin*, this program outputs DLX data for the problem of finding all solutions. (I hacked it from my old program SUDOKU-PREP, written in 2005.)

The specification consists of nine lines of ASCII characters. If the jth character of the ith line is between `1` and `9`, it defines the value that appears in cell (i,j) of the puzzle.

The puzzle is repeated in a comment line at the beginning of the output.

```
#define bufsize 16      /* beware strange behavior if you feed long lines to me! */

#include <stdio.h>
#include <stdlib.h>
  char buf[bufsize];
  int pos[9][9], row[9][9], col[9][9], box[9][9];      /* things to cover */

  main()
  {
    register int c, j, k, d, x;

    ⟨Input the given problem 2⟩;
    ⟨Output the comment line 3⟩;
    ⟨Output the item-name line 4⟩;
    ⟨Output the options 5⟩;
  }
```

2.

```
#define box(j, k) (((int)((j)/3)) ∗ 3 + (int)((k)/3))
```

⟨ Input the given problem 2 ⟩ ≡

```
  for (c = j = 0; j < 9; j++) {
    if (¬fgets(buf, bufsize, stdin)) {
      fprintf(stderr, "There␣are␣fewer␣than␣nine␣lines␣of␣input!\n");
      exit(−1);
    }
    for (k = 0; k < 9; k++)
      if (buf[k] ≥ '1' ∧ buf[k] ≤ '9') {
        d = buf[k] − '1', x = box(j, k);
        pos[j][k] = d + 1;
        if (row[j][d]) {
          fprintf(stderr, "digit␣%d␣appears␣in␣columns␣%d␣and␣%d␣of␣row␣%d!\n", d + 1,
              row[j][d] − 1, k, j);
          exit(−2);
        }
        row[j][d] = k + 1;
        if (col[k][d]) {
          fprintf(stderr, "digit␣%d␣appears␣in␣rows␣%d␣and␣%d␣of␣column␣%d!\n", d + 1,
              col[k][d] − 1, j, k);
          exit(−3);
        }
        col[k][d] = j + 1;
        if (box[x][d]) {
          fprintf(stderr, "digit␣%d␣appears␣in␣rows␣%d␣and␣%d␣of␣box␣%d!\n", d+1, box[x][d]−1, j, x);
          exit(−4);
        }
        box[x][d] = j + 1;
        c++;
      }
  }
  fprintf(stderr, "OK,␣I␣found␣%d␣clues␣in␣the␣input␣problem.\n", c);
```

This code is used in section 1.

3. ⟨ Output the comment line 3 ⟩ ≡

```
  fprintf(stdout, "|sudoku");
  for (j = 0; j < 9; j++) {
    fprintf(stdout, "!");
    for (k = 0; k < 9; k++) fprintf(stdout, "%c", pos[j][k] ? '0' + pos[j][k] : '.');
  }
  fprintf(stdout, "\n");
```

This code is used in section 1.

4. I'm going to put all `p` items first, then `r`, then `c`, then `b`, since this sort-of corresponds to the way many humans approach the subject. (And since a branch on `b` is then interesting.)

```
⟨ Output the item-name line 4 ⟩ ≡
  for (j = 0; j < 9; j++)
    for (k = 0; k < 9; k++)
      if (¬pos[j][k]) fprintf(stdout, "p%d%d␣", j, k);
  for (j = 0; j < 9; j++)
    for (k = 0; k < 9; k++)
      if (¬row[j][k]) fprintf(stdout, "r%d%d␣", j, k + 1);
  for (j = 0; j < 9; j++)
    for (k = 0; k < 9; k++)
      if (¬col[j][k]) fprintf(stdout, "c%d%d␣", j, k + 1);
  for (j = 0; j < 9; j++)
    for (k = 0; k < 9; k++)
      if (¬box[j][k]) fprintf(stdout, "b%d%d␣", j, k + 1);
  fprintf(stdout, "\n");
```

This code is used in section 1.

5. ⟨ Output the options 5 ⟩ ≡

```
  for (j = 0; j < 9; j++)
    for (k = 0; k < 9; k++)
      for (d = 0; d < 9; d++) {
        x = box(j, k);
        if (¬pos[j][k] ∧ ¬row[j][d] ∧ ¬col[k][d] ∧ ¬box[x][d])
          fprintf(stdout, "p%d%d␣r%d%d␣c%d%d␣b%d%d\n", j, k, j, d + 1, k, d + 1, x, d + 1);
      }
```

This code is used in section 1.

6. Index.

box: 1, 2, 4, 5.
buf: 1, 2.
bufsize: 1, 2.
c: 1.
col: 1, 2, 4, 5.
d: 1.
exit: 2.
fgets: 2.
fprintf: 2, 3, 4, 5.
j: 1.
k: 1.
main: 1.
pos: 1, 2, 3, 4, 5.
row: 1, 2, 4, 5.
stderr: 2.
stdin: 1, 2.
stdout: 3, 4, 5.
x: 1.

⟨ Input the given problem 2 ⟩ Used in section 1.
⟨ Output the comment line 3 ⟩ Used in section 1.
⟨ Output the item-name line 4 ⟩ Used in section 1.
⟨ Output the options 5 ⟩ Used in section 1.

SUDOKU-DLX

	Section	Page
Intro	1	1
Index	6	4