1. Intro. This is a trivial program to make data for SAT-TOMOGRAPHY. It uses the first m rows and first n columns of the image supplied on *stdin*. That image is like the ones accepted by LIFE: It has asterisks where pixels are to be 1.

It also produces an additional output file `/tmp/list`, containing '`~`*i*`x`*j*' for every asterisk in row i, column j.

```
#define mmax 100 /* max rows */
#define nmax 100 /* max columns */
#define bufsize nmax + 2 /* leave room for '\n' and '\0' */
#include <stdio.h>
#include <stdlib.h>
  char rast[mmax][nmax];
  char buf[nmax + 2];
  int m, n; /* command-line parameters */
  FILE *list_file;
  main(int argc, char *argv[])
  {
    register int d, j, k, jmax, kmax, t;
    ⟨ Check the command line 2 ⟩;
    ⟨ Open the auxiliary output file 3 ⟩;
    ⟨ Input the raster 4 ⟩;
    ⟨ Output the counts 5 ⟩;
  }
```

2. ⟨ Check the command line 2 ⟩ ≡

```
  if (argc ≠ 3 ∨ sscanf(argv[1], "%d", &m) ≠ 1 ∨ sscanf(argv[2], "%d", &n) ≠ 1) {
    fprintf(stderr, "Usage: %s m n < foo.dots > foo.tom\n", argv[0]);
    exit(−1);
  }
```

This code is used in section 1.

3. ⟨ Open the auxiliary output file 3 ⟩ ≡

```
  list_file = fopen("/tmp/list", "w");
  if (¬list_file) {
    fprintf(stderr, "I can't open `/tmp/list' for writing!\n");
    exit(−999);
  }
```

This code is used in section 1.

4. ⟨ Input the raster 4 ⟩ ≡

```
kmax = 0;
for (j = 0; j < mmax; j++) {
  if (¬fgets(buf, bufsize, stdin)) break;
  for (k = 0; k < nmax; k++) {
    if (buf[k] ≡ '\n') break;
    rast[j][k] = (buf[k] ≡ '*');
    if (rast[j][k]) fprintf(list_file, "~%dx%d\n", j + 1, k + 1);
    if (k > kmax ∧ rast[j][k]) kmax = k;
  }
}
jmax = j − 1;
fprintf(stderr, "OK, I've input an image with %d rows and %d columns.\n", jmax + 1, kmax + 1);
if (m ≤ 0 ∨ m > jmax + 1) {
  fprintf(stderr, "So your m is out of range!\n"), exit(−2);
}
if (n ≤ 0 ∨ n > kmax + 1) {
  fprintf(stderr, "So your n is out of range!\n"), exit(−3);
}
```

This code is used in section 1.

5. ⟨ Output the counts 5 ⟩ ≡

```
for (j = 0; j < m; j++) {
  for (t = 0, k = 0; k < n; k++) t += rast[j][k];
  printf("r%d=%d\n", j + 1, t);
}
for (k = 0; k < n; k++) {
  for (t = 0, j = 0; j < m; j++) t += rast[j][k];
  printf("c%d=%d\n", k + 1, t);
}
for (d = 1; d < m + n; d++) {
  for (t = 0, j = 0; j < m; j++) {
    k = d − 1 − j;
    if (k ≥ 0 ∧ k < n) t += rast[j][k];
  }
  printf("a%d=%d\n", d, t);
}
for (d = 1; d < m + n; d++) {
  for (t = 0, j = 0; j < m; j++) {
    k = j + n − d;
    if (k ≥ 0 ∧ k < n) t += rast[j][k];
  }
  printf("b%d=%d\n", d, t);
}
```

This code is used in section 1.

6. Index.

SAT-TOMOGRAPHY-PREP