1. Intro. Generate clauses for an open shop scheduling problem, as explained in the paper by Tamura, Taga, Kitagawa, and Banbara in *Constraints* **14** (2009), 254–272.

The command line contains three things: the number of machines, m; the number of jobs, n; and the desired "makespan," t.

Standard input contains an $m \times n$ matrix of work times w_{ij}, representing the time taken on machine i by job j. There are m lines of n numbers each. One or more optional title lines, each beginning with '~', may also appear at the beginning of the input; they will be echoed in the output.

The variables are ij`<`u, meaning that the starting time s_{ij} is less than u; and `!`$iji'j'$, meaning that "$s_{ij} + w_{ij} \le s_{i'j'}$ if and only if $ij < i'j'$." The latter variables appear if and only if $i = i'$ and $j \ne j'$ or $i \ne i'$ and $j = j'$ and $w_{ij} > 0$ and $w_{i'j'} > 0$.

```
#define maxmn '~' − '0'    /* jobs/machines are single characters, '0' ≤ c < '~' */
#define bufsize 128    /* for the comment lines at the beginning of stdin */
#include <stdio.h>
#include <stdlib.h>
  int m, n, t;    /* command-line parameters */
  int w[maxmn][maxmn];    /* the input matrix */
  char buf[bufsize];
  main(int argc, char *argv[])
  {
    register int i, j, ii, jj, k, l;
    ⟨ Process the command line 2 ⟩;
    ⟨ Input the matrix 3 ⟩;
    ⟨ Generate the axiom clauses 4 ⟩;
    ⟨ Generate the nonoverlap clauses 5 ⟩;
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 4 ∨ sscanf(argv[1], "%d", &m) ≠ 1 ∨ sscanf(argv[2], "%d", &n) ≠ 1 ∨ sscanf(argv[3], "%d", &t) ≠ 1)
    {
    fprintf(stderr, "Usage:␣%s␣m␣n␣t␣<␣w[m][n]\n", argv[0]);
    exit(−1);
  }
  if (m > maxmn) {
    fprintf(stderr, "Sorry,␣m␣(%d)␣must␣not␣exceed␣%d!\n", m, maxmn);
    exit(−2);
  }
  if (n > maxmn) {
    fprintf(stderr, "Sorry,␣n␣(%d)␣must␣not␣exceed␣%d!\n", n, maxmn);
    exit(−3);
  }
```

This code is used in section 1.

3. I don't do any fancy error checking about breaks between lines.

```
⟨ Input the matrix 3 ⟩ ≡
  while (1) {
    i = getc(stdin); ungetc(i, stdin);
    if (i ≠ '~') break;
    fgets(buf, bufsize, stdin);
    printf("%s", buf);
  }
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
      if (fscanf(stdin, "%d", &w[i][j]) ≠ 1) {
        fprintf(stderr, "Oops,␣I␣had␣trouble␣reading␣w%d%d!\n", i, j);
        exit(−4);
      }
      if (w[i][j] < 0 ∨ w[i][j] > t) {
        fprintf(stderr, "Oops,␣w%d%d␣should␣be␣between␣0␣and␣%d,␣not␣%d!\n", i, j, t, w[i][j]);
        exit(−5);
      }
    }
  }
  for (i = 0; i < m; i++) {
    for (k = 0, j = 0; j < n; j++) k += w[i][j];
    if (k > t) {
      fprintf(stderr, "Unsatisfiable␣(machine␣%d␣needs␣%d)!\n", i, k);
      exit(−10);
    }
  }
  for (j = 0; j < n; j++) {
    for (k = 0, i = 0; i < m; i++) k += w[i][j];
    if (k > t) {
      fprintf(stderr, "Unsatisfiable␣(job␣%d␣needs␣%d)!\n", j, k);
      exit(−11);
    }
  }
  printf("~␣sat-oss␣%d␣%d␣%d\n", m, n, t);
  for (i = 0; i < m; i++) {
    printf("~␣");
    for (j = 0; j < n; j++) printf("%4d", w[i][j]);
    printf("\n");
  }
```

This code is used in section 1.

4. The starting time s_{ij} will be at most $t - w_{ij}$. We don't assign starting times when $w_{ij} = 0$; such times can always be assumed to be 0 without loss of generality.

```
⟨ Generate the axiom clauses 4 ⟩ ≡
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      if (w[i][j])
        for (l = 1; l < t − w[i][j]; l++)
          printf("~%c%c<%d␣%c%c<%d\n", '0' + i, '0' + j, l, '0' + i, '0' + j, l + 1);
```

This code is used in section 1.

5. ⟨ Generate the nonoverlap clauses 5 ⟩ ≡

```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    if (w[i][j]) {
      for (ii = 0; ii < m; ii++)
        for (jj = 0; jj < n; jj++)
          if (((ii ≡ i ∧ jj ≠ j) ∨ (ii ≠ i ∧ jj ≡ j)) ∧ w[ii][jj]) {
            for (l = 0; l + w[i][j] ≤ t + 1 − w[ii][jj]; l++) {
              if (i < ii ∨ j < jj) printf("~!%c%c%c%c", '0' + i, '0' + j, '0' + ii, '0' + jj);
              else printf("!%c%c%c%c", '0' + ii, '0' + jj, '0' + i, '0' + j);
              if (l > 0) printf("␣%c%c<%d", '0' + i, '0' + j, l);
              if (l + w[i][j] < t + 1 − w[ii][jj]) printf("␣~%c%c<%d", '0' + ii, '0' + jj, l + w[i][j]);
              printf("\n");
            }
          }
    }
```

This code is used in section 1.

6. Index.

SAT-OSS