1. Intro. This quickie outputs clauses that are unsatisfiable because they state that there exists a partial ordering on m elements in which no element is maximal. (All backtrack proofs of this fact are known to require $\Omega(2^m)$ steps.)

Variable $j\,.\,k$ means that $j \prec k$ in the partial ordering.

```
#include <stdio.h>
#include <stdlib.h>
  int m;

  main(int argc, char *argv[])
  {
    register i, j, k;
    ⟨ Process the command line 2 ⟩;
    ⟨ Generate the clauses for irreflexivity 3 ⟩;
    ⟨ Generate the clauses for transitivite 4 ⟩;
    ⟨ Generate the clauses for nonmaximality 5 ⟩;
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 2 ∨ sscanf(argv[1], "%d", &m) ≠ 1) {
    fprintf(stderr, "Usage:␣%s␣m\n", argv[0]);
    exit(−1);
  }
  printf("~␣sat-poset-nomax␣%d\n", m);
```

This code is used in section 1.

3. ⟨ Generate the clauses for irreflexivity 3 ⟩ ≡

```
  for (j = 1; j ≤ m; j++) printf("~%d.%d\n", j, j);
```

This code is used in section 1.

4. ⟨ Generate the clauses for transitivite 4 ⟩ ≡

```
  for (i = 1; i ≤ m; i++)
    for (j = 1; j ≤ m; j++)
      if (i ≠ j) {
        for (k = 1; k ≤ m; k++)
          if (j ≠ k) {
            printf("~%d.%d␣~%d.%d␣%d.%d\n", i, j, j, k, i, k);
          }
      }
```

This code is used in section 1.

5. ⟨ Generate the clauses for nonmaximality 5 ⟩ ≡

```
  for (j = 1; j ≤ m; j++) {
    for (k = 1; k ≤ m; k++) printf("␣%d.%d", j, k);
    printf("\n");
  }
```

This code is used in section 1.

6. Index.

argc: 1, 2.
argv: 1, 2.
exit: 2.
fprintf: 2.
i: 1.
j: 1.
k: 1.
m: 1.
main: 1.
printf: 2, 3, 4, 5.
sscanf: 2.
stderr: 2.

⟨ Generate the clauses for irreflexivity 3 ⟩ Used in section 1.
⟨ Generate the clauses for nonmaximality 5 ⟩ Used in section 1.
⟨ Generate the clauses for transitivite 4 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.

SAT-POSET-NOMAX

	Section	Page
Intro	1	1
Index	6	2