**1. Intro.** Generate SAT instances for van der Waerden configurations: There are to be no $k$ equally spaced zeroes and no $l$ equally spaced ones, in a binary string of length $n$. (If unsatisfiable, every red-green coloring of $\{1, 2, \ldots, n\}$ contains a red arithmetic progression of length $k$ or a green arithmetic progression of length $l$.)

The numbers $k$, $l$, $n$ are given on the command line.

```
#include <stdio.h>
#include <stdlib.h>
  int k, l, n;
  main(int argc, char *argv[])
  {
    register int i, j, d;
    ⟨ Process the command line 2 ⟩;
    ⟨ Output the positive clauses 3 ⟩;
    ⟨ Output the negative clauses 4 ⟩;
  }
```

**2.** ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 4 ∨ sscanf(argv[1], "%d", &k) ≠ 1 ∨ sscanf(argv[2], "%d", &l) ≠ 1 ∨ sscanf(argv[3], "%d", &n) ≠ 1)
    {
    fprintf(stderr, "Usage:␣%s␣k␣l␣n\n", argv[0]);
    exit(−1);
  }
  printf("~␣sat-waerden␣%d␣%d␣%d\n", k, l, n);
```

This code is used in section 1.

**3.** ⟨ Output the positive clauses 3 ⟩ ≡

```
  for (d = 1; 1 + (k − 1) * d ≤ n; d++) {
    for (i = 1; i + (k − 1) * d ≤ n; i++) {
      for (j = 0; j < k; j++) printf("␣%d", i + j * d);
      printf("\n");
    }
  }
```

This code is used in section 1.

**4.** ⟨ Output the negative clauses 4 ⟩ ≡

```
  for (d = 1; 1 + (l − 1) * d ≤ n; d++) {
    for (i = 1; i + (l − 1) * d ≤ n; i++) {
      for (j = 0; j < l; j++) printf("␣~%d", i + j * d);
      printf("\n");
    }
  }
```

This code is used in section 1.

### 5. Index.

⟨ Output the negative clauses 4 ⟩ Used in section 1.
⟨ Output the positive clauses 3 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.

# SAT-WAERDEN