1. Intro. This program generates clauses that enforce the constraint $x_1 + \cdots + x_n \le r$, using a method due to Carsten Sinz [*Lecture Notes in Computer Science* **3709** (2005), 827–831]. It introduces $r(n-r)$ new variables $\texttt{S}i\texttt{.}j$ for $1 \le i \le n-r$ and $1 \le j \le r$, and generates a total of $(r+1)(n-r)+r(n-r-1)$ clauses involving these variables and x_1 through x_n. All clauses have length 3 or less.

With change files we can change the names of the variables x_i.

```
#include <stdio.h>
#include <stdlib.h>
  int n, r;      /* the given parameters */
  main(int argc, char *argv[])
  {
    register int i, j, k;
    ⟨ Process the command line 2 ⟩;
    for (j = 1; j ≤ r; j++) ⟨ Generate the horizontal clauses for row j 3 ⟩;
    for (j = 0; j ≤ r; j++) ⟨ Generate the vertical clauses for row j 4 ⟩;
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 3 ∨ sscanf(argv[1], "%d", &n) ≠ 1 ∨ sscanf(argv[2], "%d", &r) ≠ 1) {
    fprintf(stderr, "Usage: %s n r\n", argv[0]);
    exit(-1);
  }
  if (r < 0 ∨ r ≥ n) {
    fprintf(stderr, "Eh? r should be between 0 and n-1!\n");
    exit(-2);
  }
  printf("~ sat-threshold-sinz %d %d\n", n, r);
```

This code is used in section 1.

3. ⟨ Generate the horizontal clauses for row j 3 ⟩ ≡

```
  for (i = 1; i < n - r; i++) printf("~S%d.%d S%d.%d\n", i, j, i + 1, j);
```

This code is used in section 1.

4.

```
#define xbar(k) printf("~x%d", k)
```

⟨ Generate the vertical clauses for row j 4 ⟩ ≡

```
  for (i = 1; i ≤ n - r; i++) {
    xbar(i + j);
    if (j) printf(" ~S%d.%d", i, j);
    if (j < r) printf(" S%d.%d", i, j + 1);
    printf("\n");
  }
```

This code is used in section 1.

5. Index.

⟨ Generate the horizontal clauses for row j 3 ⟩ Used in section 1.
⟨ Generate the vertical clauses for row j 4 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.

SAT-THRESHOLD-SINZ