1. Intro. Generate SAT instances for Erdős discrepancy patterns: The sequences $(x_d, x_{2d}, \ldots, x_{\lfloor n/d \rfloor d})$ are supposed to be strongly balanced, for $1 \le d \le n$, where a sequence $(y_1, \ldots, y_t)$ is "strongly balanced" if the corresponding sequence of ± 1s defined by $z_j = 2y_j - 1$ has all partial sums satisfying $-2 \le z_1 + \cdots + z_k \le 2$.

It's easy to see that the latter property needs to be checked only for odd values of k with $3 \le k \le t$.

```
#include <stdio.h>
#include <stdlib.h>
  int n;
  ⟨ Subroutine 3 ⟩
  main(int argc, char *argv[])
  {
    register int d;
    ⟨ Process the command line 2 ⟩;
    printf("~␣sat-erdos-disc␣%d\n", n);
    printf("X%d\n", n < 720 ? 360 : 720);    /* might as well save a factor of two */
    for (d = 1; 3 * d ≤ n; d++) generate(d, n/d);
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

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

This code is used in section 1.

3. Our task is to generate clauses that characterize a strongly balanced sequence, and it turns out that there's a very interesting way to do this. The subroutine *generate*(d,n) makes clauses for the sequence with $y_j = x_{jd}$.

Sinz's cardinality clauses (see *TAOCP* Section 7.2.2.2) have the property that $y_1+\cdots+y_{j+k-1} \geq k$ implies S_j^k; hence we want $S_j^{j+2} = 0$ for $j < n/2$. The dual clauses have the property that $\bar{y}_1 + \cdots + \bar{y}_{j+k-1} \geq k$ implies $\bar{S}_k^j$; we can rewrite this to say that S_k^j implies $y_1 + \cdots + y_{j+k-1} \geq j$. Hence we also want $S_{k+2}^k = 1$ for $k < n/2$. It follows that we need only deal with auxiliary variables S_j^k when $|j-k| \leq 1$. The variables S_k^{k-1}, S_k^k, and S_k^{k+1} will be denoted respectively by dAk, dBk, and dCk.

The clauses

$$(\bar{S}_t^t \vee S_{t+1}^t) \wedge (\bar{S}_t^{t+1} \vee S_{t+1}^{t+1}) \wedge (S_t^t \vee \bar{S}_t^{t+1}) \wedge (S_{t+1}^t \vee \bar{S}_{t+1}^{t+1})$$

are needed when $n \geq 2t+3$. The clauses

$$(\bar{y}_{2t-2} \vee S_t^{t-1}) \wedge (\bar{y}_{2t-1} \vee \bar{S}_t^{t-1} \vee S_t^t) \wedge (\bar{y}_{2t} \vee \bar{S}_t^t \vee S_t^{t+1}) \wedge (\bar{y}_{2t+1} \vee \bar{S}_t^{t+1})$$

and their duals

$$(y_{2t-2} \vee \bar{S}_{t-1}^t) \wedge (y_{2t-1} \vee S_{t-1}^t \vee \bar{S}_t^t) \wedge (y_{2t} \vee S_t^t \vee \bar{S}_{t+1}^t) \wedge (y_{2t+1} \vee S_{t+1}^t)$$

are needed when $n \geq 2t+1$. (And we simplify these clauses for small t by using the facts that $S_j^0 = 1$ and $S_0^k = 0$.)

```
⟨ Subroutine 3 ⟩ ≡
  void generate(int d, int n)
  {
    register int i, j, k, t;
    for (t = 1; 2 * t + 3 ≤ n; t++) ⟨ Generate the first clauses 4 ⟩;
    for (t = 1; 2 * t + 1 ≤ n; t++) ⟨ Generate the second clauses 5 ⟩;
  }
```

This code is used in section 1.

4. ⟨ Generate the first clauses 4 ⟩ ≡

```
  {
    printf("~%dB%d␣%dA%d\n", d, t, d, t + 1);
    printf("~%dC%d␣%dB%d\n", d, t, d, t + 1);
    printf("%dB%d␣~%dC%d\n", d, t, d, t);
    printf("%dA%d␣~%dB%d\n", d, t + 1, d, t + 1);
  }
```

This code is used in section 3.

5. ⟨ Generate the second clauses 5 ⟩ ≡

```
{
  if (t > 1) {
    printf("~X%d␣%dA%d\n", d * (t + t - 2), d, t);
    printf("X%d␣~%dC%d\n", d * (t + t - 2), d, t - 1);
    printf("~X%d␣~%dA%d␣%dB%d\n", d * (t + t - 1), d, t, d, t);
    printf("X%d␣%dC%d␣~%dB%d\n", d * (t + t - 1), d, t - 1, d, t);
  }
  else {
    printf("~X%d␣%dB%d\n", d, d, 1);
    printf("X%d␣~%dB%d\n", d, d, 1);
  }
  printf("~X%d␣~%dB%d␣%dC%d\n", d * (t + t), d, t, d, t);
  printf("X%d␣%dB%d␣~%dA%d\n", d * (t + t), d, t, d, t + 1);
  printf("~X%d␣~%dC%d\n", d * (t + t + 1), d, t);
  printf("X%d␣%dA%d\n", d * (t + t + 1), d, t + 1);
}
```

This code is used in section 3.

6. Index.

⟨ Generate the first clauses 4 ⟩ Used in section 3.
⟨ Generate the second clauses 5 ⟩ Used in section 3.
⟨ Process the command line 2 ⟩ Used in section 1.
⟨ Subroutine 3 ⟩ Used in section 1.

SAT-ERDOS-DISC