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 \(1 \leq d \leq n\), where a sequence \((y_1, \ldots, y_t)\) is “strongly balanced” if the corresponding sequence of \(\pm 1\)s defined by \(z_j = 2y_j - 1\) has all partial sums satisfying \(-2 \leq z_1 + \cdots + z_k \leq 2\).

It’s easy to see that the latter property needs to be checked only for odd values of \(k\) with \(3 \leq k \leq t\).

```c
#include <stdio.h>
#include <stdlib.h>

int n;

(Subroutine 3)

int 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: sat-erdos-disc n", argv[0]);
    exit(-1);
}
```

This code is used in section 1.
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 $\text{generate}(d,n)$ 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 $\bar{S}_j^k$; we can rewrite this to say that $S_j^k$ 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 only need deal with auxiliary variables $S_{k,j}$ when $j \leq k$.

The variables $S_{k,1}$, $S_{k,k}$, and $S_{k,k+1}$ will be denoted respectively by $dA_k$, $dB_k$, and $dC_k$.

The clauses

$$(S_t^i \lor S_{t+1}^i) \land (S_{t+1}^i \lor S_{t+1}^{i+1}) \land (S_t^i \lor S_t^{i+1}) \land (S_t^i \lor S_{t+1}^{i+1})$$

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

$$(\bar{y}_{2t-2} \lor S_{t-1}^i) \land (\bar{y}_{2t-1} \lor S_{t-1}^i \lor S_t^i) \land (\bar{y}_{2t} \lor S_t^i \lor S_{t+1}^i) \land (\bar{y}_{2t+1} \lor S_{t+1}^i)$$

and their duals

$$(y_{2t-2} \lor S_{t-1}^i) \land (y_{2t-1} \lor S_{t-1}^i \lor S_t^i) \land (y_{2t} \lor S_t^i \lor S_{t+1}^i) \land (y_{2t+1} \lor S_{t+1}^i)$$

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

(Subtitle 3) $\equiv$

```c
void generate(int d, int n)
{
    register int i, j, k, t;
    for (t = 1; 2 * t + 3 \leq n; t++) { Generate the first clauses 4; }
    for (t = 1; 2 * t + 1 \leq n; t++) { Generate the second clauses 5; }
}
```
This code is used in section 1.

4. (Generate the first clauses 4) $\equiv$

```c
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) \( \equiv \)

\[
\begin{align*}
\text{if} & \ (t > 1) \{ \\
& \quad \text{printf}("~X%d␣\%dA%d\n", \ d * (t + t - 2), \ d, \ t); \\
& \quad \text{printf}("X%d␣\~\%dC%d\n", \ d * (t + t - 2), \ d, \ t - 1); \\
& \quad \text{printf}("\~X%d␣\~\%dA%d\%dB%d\n", \ d * (t + t - 1), \ d, \ d, \ t); \\
& \quad \text{printf}("X%d\%dC%d\~\%dB%d\n", \ d * (t + t - 1), \ d, \ t - 1, \ d, \ t); \\
\} \\
\text{else} & \{ \\
& \quad \text{printf}("\~X%d\%dB%d\n", \ d, \ d, \ 1); \\
& \quad \text{printf}("X%d\~\%dB%d\n", \ d, \ d, \ 1); \\
\} \\
& \quad \text{printf}("\~X%d\~\%dB%d\%dC%d\n", \ d * (t + t), \ d, \ t, \ d, \ t, \ t); \\
& \quad \text{printf}("X%d\%dB%d\~\%dA%d\n", \ d * (t + t), \ d, \ d, \ t, \ t + 1); \\
& \quad \text{printf}("\~X%d\~\%dB%d\n", \ d * (t + t + 1), \ d, \ t); \\
& \quad \text{printf}("X%d\~\%dB%d\n", \ d * (t + t + 1), \ d, \ t + 1); \\
\}
\end{align*}
\]

This code is used in section 3.
6. Index.

`argc`: 1, 2.
`argv`: 1, 2.
`d`: 1, 3.
`exit`: 2.
`fprintf`: 2.
`generate`: 1, 3.
`i`: 3.
`j`: 3.
`k`: 3.
`main`: 1.
`n`: 1, 3.
`printf`: 1, 4, 5.
`sscanf`: 2.
`stderr`: 2.
`t`: 3.
(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.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro</td>
<td>1</td>
</tr>
<tr>
<td>Index</td>
<td>6</td>
</tr>
</tbody>
</table>