1. Intro. This little program generates random data for SAT-CLOSEST-STRING. It takes five parameters from the command line:

- n, the length of strings.
- m, the number of strings.
- $r_{\min}$ and $r_{\max}$, bounds on the distances r_j.
- s, the random seed.

```
#define maxn 10000
#include <stdio.h>
#include <stdlib.h>
#include "gb_flip.h"
  char secret[maxn + 1];      /* a solution (probably the only one) */
  int del[maxn];      /* timestamps for divergence from the secret */
  int m, n, rmin, rmax, s;      /* command-line parameters */
  main(int argc, char *argv[])
  {
    register int c, i, j, k, o, r;
    ⟨ Process the command line 2 ⟩;
    ⟨ Generate the secret key 3 ⟩;
    for (c = o = 0, j = 1; j ≤ m; j++) ⟨ Generate s_j and r_j 4 ⟩;
    fprintf(stderr, "OK, I generated %d strings, with %d collisions and %d overlaps.\n", m, c, o);
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 6 ∨ sscanf(argv[1], "%d", &n) ≠ 1 ∨ sscanf(argv[2], "%d", &m) ≠ 1 ∨ sscanf(argv[3], "%d",
        &rmin) ≠ 1 ∨ sscanf(argv[4], "%d", &rmax) ≠ 1 ∨ sscanf(argv[5], "%d", &s) ≠ 1) {
    fprintf(stderr, "Usage: %s n m rmin rmax seed\n", argv[0]);
    exit(−1);
  }
  if (n ≡ 0 ∨ n > maxn) {
    fprintf(stderr, "Oops: n should be between 1 and %d, not %d!\n", maxn, n);
    exit(−2);
  }
  if (rmin ≤ 0 ∨ rmin > rmax ∨ rmax ≥ n) {
    fprintf(stderr, "Oops: I assume that 0 < rmin <= rmax < n!\n");
    exit(−3);
  }
  printf("! sat-closest-string-dat %d %d %d %d %d\n", n, m, rmin, rmax, s);
  gb_init_rand(s);
```

This code is used in section 1.

3. ⟨ Generate the secret key 3 ⟩ ≡

```
  for (i = 0; i < n; i++) secret[i] = gb_unif_rand(2) + '0';
  printf("! %s\n", secret);
```

This code is used in section 1.

4. ⟨ Generate s_j and r_j 4 ⟩ ≡

```
{
  r = rmin;
  if (r < rmax) r += gb_unif_rand(rmax − rmin + 1);
  for (k = 0; k < r; k++) {
    i = gb_unif_rand(n);
    if (del[i] ≡ j) c++;      /* c counts collisions within s_j */
    else if (del[i]) o++;     /* o counts overlaps with previous cases */
    del[i] = j;
  }
  for (i = 0; i < n; i++) printf("%c", del[i] ≡ j ? secret[i] ⊕ 1 : secret[i]);
  printf("␣%d\n", r);
}
```

This code is used in section 1.

5. Index.

⟨ Generate s_j and r_j 4 ⟩ Used in section 1.
⟨ Generate the secret key 3 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.

SAT-CLOSEST-STRING-DAT