

(Downloaded from <https://cs.stanford.edu/~knuth/programs.html> and typeset on September 17, 2017)

1. One-dimensional particle physics. This program is a quick-and-dirty implementation of the random process analyzed by Hermann Rost in 1981 (see exercise 5.1.4–40). Start with infinitely many 1s followed by infinitely many 0s; then randomly interchange adjacent elements that are out of order.

```
#include <stdio.h>
#include <math.h>
#include "gb_flip.h"
char *bit;
int *list;
int seed; /* random number seed */
int n; /* this many interchanges */
main(argc, argv)
    int argc;
    char *argv[];
{
    register int i, j, k, l, t, u, r;
    < Scan the command line 2 >;
    < Initialize everything 3 >;
    for (r = 0; r < n; r++) < Move 4 >;
    < Print the results 5 >;
}
```

2. < Scan line 2 > \equiv
 if ($argc \neq 3 \vee sscanf(argv[1], "%d", &n) \neq 1 \vee sscanf(argv[2], "%d", &seed) \neq 1$) {
 fprintf(stderr, "Usage: %s %d %d\n", argv[0], n, seed);
 exit(-1);
 }

This code is used in section 1.

3. We maintain the following invariants: $bit[k] = 1$ for $k \leq l$; $bit[k] = 0$ for $k = u$; the indices i where $bit[i] > bit[i + 1]$ are $list[j]$ for $0 \leq j < t$.

```
< Initialize everything 3 >  $\equiv$ 
gb_init_rand(seed);
bit = (char *) malloc(2 * n + 2);
list = (int *) malloc(4 * n + 4);
for (k = 0; k <= n; k++) bit[k] = 1;
for ( ; k <= n + n + 1; k++) bit[k] = 0;
l = u = n;
list[0] = n;
t = 1;
```

This code is used in section 1.

4. $\langle \text{Move } 4 \rangle \equiv$

```
{
  j = gb_unif_rand(t);
  i = list[j];
  t--;
  list[j] = list[t];
  bit[i] = 0; bit[i + 1] = 1;
  if (i ≡ l) l--;
  if (i ≡ u) u++;
  if (bit[i - 1]) list[t++] = i - 1;
  if (¬bit[i + 2]) list[t++] = i + 1;
}
```

This code is used in section 1.

5. $\langle \text{Print the results } 5 \rangle \equiv$

```
 $\langle \text{Print the PostScript header info } 6 \rangle$ ;
 $\langle \text{Print the empirical curve } 8 \rangle$ ;
 $\langle \text{Print the theoretical curve } 9 \rangle$ ;
 $\langle \text{Print the PostScript trailer info } 7 \rangle$ ;
```

This code is used in section 1.

6. $\langle \text{Print the PostScript header info } 6 \rangle \equiv$

```
printf("%%!PS\n");
printf("%%BoundingBox:-1-1 361 361\n");
printf("%%Creator: %s %s\n", argv[0], argv[1], argv[2]);
printf("/d_{0 s neg rlineto} bind def\n"); /* move down */
printf("/r_{s 0 rlineto} bind def\n"); /* move right */
```

This code is used in section 5.

7. $\langle \text{Print the PostScript trailer info } 7 \rangle \equiv$

```
printf("showpage\n");
```

This code is used in section 5.

8. The empirical curve is scaled so that $\sqrt{6n}$ units is 5 inches.

$\langle \text{Print the empirical curve } 8 \rangle \equiv$

```
printf("/s %g def\n", 360.0/sqrt(6.0 * n));
printf("newpath %d %d s mul moveto\n", 0, n - l);
for (k = l + 1; k ≤ u; k++) {
  if (bit[k]) printf("_d"); else printf("_r");
  if ((k - l) % 40 ≡ 0) printf("\n");
}
printf("\n0 0 lineto closepath\n");
printf("1 setlinewidth stroke\n");
```

This code is used in section 5.

9. The theoretical curve $\sqrt{x} + \sqrt{y} = 1$ is scaled so that 1 unit is 5 inches. We use the fact that this curve is *exactly* drawn by PostScript's Bezier curve routines, from the control points (0, 1), (0, 1/3), (1/3, 0), (1, 0).

$\langle \text{Print the theoretical curve } 9 \rangle \equiv$

```
printf("newpath 0 360 moveto 0 120 120 0 360 0 curveto\n");
printf("0 0 0 lineto closepath\n");
printf(".3 setlinewidth stroke\n");
```

This code is used in section 5.

10. Index.

argc: [1](#), [2](#).
argv: [1](#), [2](#), [6](#).
bit: [1](#), [3](#), [4](#), [8](#).
exit: [2](#).
fprintf: [2](#).
gb_init_rand: [3](#).
gb_unif_rand: [4](#).
i: [1](#).
j: [1](#).
k: [1](#).
l: [1](#).
list: [1](#), [3](#), [4](#).
main: [1](#).
malloc: [3](#).
n: [1](#).
printf: [6](#), [7](#), [8](#), [9](#).
r: [1](#).
seed: [1](#), [2](#), [3](#).
sqr: [8](#).
sscanf: [2](#).
stderr: [2](#).
t: [1](#).
u: [1](#).

⟨ Initialize everything [3](#) ⟩ Used in section [1](#).
⟨ Move [4](#) ⟩ Used in section [1](#).
⟨ Print the PostScript header info [6](#) ⟩ Used in section [5](#).
⟨ Print the PostScript trailer info [7](#) ⟩ Used in section [5](#).
⟨ Print the empirical curve [8](#) ⟩ Used in section [5](#).
⟨ Print the results [5](#) ⟩ Used in section [1](#).
⟨ Print the theoretical curve [9](#) ⟩ Used in section [5](#).
⟨ Scan the command line [2](#) ⟩ Used in section [1](#).

ROST

	Section	Page
One-dimensional particle physics	1	1
Index	10	3