1. Intro. A simple program to make "random" squaregraphs, by sort of a "crocheting" technique. (Hacked in haste.)

```
#define maxn 1000
#include <stdio.h>
#include <stdlib.h>
#include "gb_flip.h"
  int a[2 * maxn + 4], d[2 * maxn + 8];
  int move[8 * maxn];
  int count[maxn];
  int seed;
  int steps;
  main(int argc, char *argv[])
  {
    register int j, k, m, t, w;
    ⟨ Process the command line 2 ⟩;
    a[0] = 0, a[1] = 1, a[2] = 0, a[3] = 1;
    d[0] = d[1] = d[2] = d[3] = 2;
    w = 4;
    for (j = 0; j < steps; j++) {
      ⟨ Set m to the number of possible moves 3 ⟩;
      k = gb_unif_rand(m);
      ⟨ Make move k 4 ⟩;
      ⟨ Check for pairs 5 ⟩;
    }
    ⟨ Output the result 6 ⟩;
  }
```

2. ⟨ Process the command line 2 ⟩ ≡

```
  if (argc ≠ 3 ∨ sscanf(argv[1], "%d", &steps) ≠ 1 ∨ sscanf(argv[2], "%d", &seed) ≠ 1) {
    fprintf(stderr, "Usage: %s n seed\n", argv[0]);
    exit(-1);
  }
  if (steps ≥ maxn) {
    fprintf(stderr, "Sorry, n should be less than %d!\n", maxn);
    exit(-2);
  }
  gb_init_rand(seed);
```

This code is used in section 1.

3. ⟨ Set m to the number of possible moves 3 ⟩ ≡

```
  d[w] = d[0], d[w + 1] = d[1], a[w] = a[0], a[w + 1] = a[1];
  for (m = 0; m < w; m++) move[m] = m;
  for (k = 0; k < w; k++)
    if (d[k + 1] > 3) move[m++] = maxn + k;
  for (k = 0; k < w; k++)
    if (d[k + 1] > 3 ∧ d[k + 2] > 3) move[m++] = maxn + maxn + k;
```

This code is used in section 1.

4. ⟨ Make move k 4 ⟩ ≡

```
if (move[k] < maxn) {
  w += 2, k = move[k];
  for (m = w − 1; m ≥ k + 2; m−−) d[m + 2] = d[m], a[m + 2] = a[m];
  d[k + 3] = d[k + 1] + 1, d[k + 2] = d[k + 1] = 2, d[k] = d[k] + 1;
  a[k + 3] = a[k + 1], a[k + 2] = j + 2, a[k + 1] = a[k], a[k] = j + 2;
  if (k + 3 ≥ w)
    for (t = 0; t + w ≤ k + 3; t++) d[t] = d[w + t], a[t] = a[w + t];
} else if (move[k] < maxn + maxn) {
  k = move[k] − maxn;
  d[k + 1] = 2;
  t = a[k + 1], a[k + 1] = a[k], a[k] = t;
  if (k + 1 ≥ w)
    for (t = 0; t + w ≤ k + 1; t++) d[t] = d[w + t], a[t] = a[w + t];
} else {
  k = move[k] − maxn − maxn;
  for (t = 0; t < w; t++)
    if (a[t] ≡ a[k + 2] ∧ t ≠ k + 2) a[t] = a[k];
  a[k] = a[k + 1], a[k + 1] = a[k + 3], d[k] = d[k] + 1, d[k + 1] = d[k + 3] + 1;
  w −= 2;
  for (t = k + 2; t < w; t++) a[t] = a[t + 2], d[t] = d[t + 2];
}
```

This code is used in section 1.

5. ⟨ Check for pairs 5 ⟩ ≡

```
for (k = 0; k < j + 2; k++) count[k] = 0;
for (k = 0; k < w; k++) count[a[k]]++;
for (k = 0; k < j + 2; k++)
  if (count[k] ≠ 0 ∧ count[k] ≠ 2) fprintf(stderr, "count[%d] is %d!\n", k, count[k]);
```

This code is used in section 1.

6. ⟨ Output the result 6 ⟩ ≡

```
for (k = 0; k < w; k++) {
  printf(" %d", a[k]);
  if (k % 20 ≡ 19) printf("\n");
}
if (k % 20 ≠ 0) printf("\n");
```

This code is used in section 1.

7. Index.

SQUAREGRAPH-RAND