1. Intro. Given a nonempty parade on the command line, this quick-and-dirty program computes its "recursive rank," as explained in my unpublication *Parades and poly-Bernoulli bijections*.

The rank might be huge. So I don't actually compute it; I produce Mathematica code that will do the numerical work.

(Sorry — I hacked this up in a huge hurry.)

```
#define maxn 100

#include <stdio.h>
#include <stdlib.h>
  int strg[maxn], strb[maxn];    /* the digit strings */
  int d;    /* the order */
  int m, n;    /* how many girls and boys? */
  int name[maxn];    /* original names of the current boys */
  int x[maxn], xname[maxn];    /* the type */

  main(int argc, char *argv[])
  {
    register int i, j, k, prevj, t, p, l, max;

    ⟨Process the command line 2⟩;
    ⟨Print the boilerplate to get Mathematica started 5⟩;
    for (j = 1; j ≤ n; j++) name[j] = j;
    while (m) ⟨Remove girl m and reduce the parade 3⟩;
    printf("0\n");    /* finish the Mathematica code */
  }
```

2. An incorrect command line aborts the run. But we do explain what was wrong.

```
⟨ Process the command line 2 ⟩ ≡
  if (argc < 2) {
    fprintf(stderr, "Usage:␣%s␣<parade>\n", argv[0]);
    exit(−1);
  }
  for (k = 1; k < maxn; k++) strg[k] = strb[k] = −1;
  for (d = 0, k = 1; argv[k]; k++) {
    if (argv[k][0] ≠ 'g' ∧ argv[k][0] ≠ 'b') {
      fprintf(stderr, "Bad␣argument␣‘%s’;␣should␣start␣with␣g␣or␣b!\n", argv[k]);
      exit(−2);
    }
    for (prevj = j, j = 0, i = 1; argv[k][i] ≥ '0' ∧ argv[k][i] ≤ '9'; i++) j = 10 ∗ j + argv[k][i] − '0';
    if (j ≡ 0 ∨ argv[k][i]) {
      fprintf(stderr, "Bad␣argument␣‘%s’;␣should␣be␣a␣positive␣number!\n", argv[k]);
      exit(−3);
    }
    if (j ≥ maxn) {
      fprintf(stderr, "Recompile␣me:␣maxn=%d!\n", maxn);
      exit(−6);
    }
    if (argv[k][0] ≡ 'g' ∧ j > m) m = j;
    else if (argv[k][0] ≡ 'b' ∧ j > n) n = j;
    if ((argv[k][0] ≡ 'g' ∧ strg[j] ≥ 0) ∨ (argv[k][0] ≡ 'b' ∧ strb[j] ≥ 0)) {
      fprintf(stderr, "You've␣already␣mentioned␣%s!\n", argv[k]);
      exit(−4);
    }
    if (argv[k][0] ≡ argv[k − 1][0] ∧ prevj > j) {
      fprintf(stderr, "Out␣of␣order:␣%s>%s!\n", argv[k − 1], argv[k]);
      exit(−5);
    }
    if (argv[k][0] ≡ 'b' ∧ argv[k − 1][0] ≠ 'b') d++;
    if (argv[k][0] ≡ 'g') strg[j] = d; else strb[j] = d;
  }
  if (argv[k − 1][0] ≡ 'b') {      /∗ parade ended with a boy: d is too large ∗/
    d−−;      /∗ however I still keep the entry d + 1, not 0, in strb! ∗/
  }
  for (j = 1; j ≤ m; j++)
    if (strg[j] < 0) {
      fprintf(stderr, "girl␣g%d␣is␣missing!\n", j);
      exit(−7);
    }
  for (j = 1; j ≤ n; j++)
    if (strb[j] < 0) {
      fprintf(stderr, "boy␣b%d␣is␣missing!\n", j);
      exit(−8);
    }
  fprintf(stderr, "OK,␣that's␣a␣valid␣parade␣of␣order␣%d␣with␣%d␣girls␣and␣%d␣boys!\n", d, m, n);
```

This code is used in section 1.

3. ⟨Remove girl *m* and reduce the parade 3⟩ ≡

```
{
  t = strg[m] + 1;    /* boys in block t = current type */
  for (max = n; max; max--)
    if (strb[max] ≡ t) break;
  if (max ≡ 0) l = 0, p = n + 1;
  else {
    for (l = 0, p = j = 1; j ≤ n; j++) {
      if (strb[j] ≡ t ∧ j ≠ max) x[l] = j, xname[l++] = name[j];
      else strb[p] = strb[j], name[p++] = name[j];
    }
    x[l] = max, xname[l] = name[max − l], l++;
    ⟨Renumber the blocks if block t is going away 4⟩;
  }
  ⟨Report what we just did 6⟩;
  n = p − 1, m--;
}
```

This code is used in section 1.

4. ⟨Renumber the blocks if block *t* is going away 4⟩ ≡

```
if (t > 1) {
  for (j = 1; j < m; j++)
    if (strg[j] ≡ t − 1) break;
  if (j ≡ m) {    /* block t joins block t − 1 */
    for (j = 1; j < m; j++)
      if (strg[j] ≥ t) strg[j]--;
    for (j = 1; j < p; j++)
      if (strb[j] ≥ t) strb[j]--;
    t--, d--;
  }
}
```

This code is used in section 3.

5. ⟨Print the boilerplate to get Mathematica started 5⟩ ≡

```
printf("(*␣output␣from␣%s", argv[0]);
for (k = 1; argv[k]; k++) printf("␣%s", argv[k]);
printf("␣*)\n");
printf("b=Binomial\n");
printf("brank[typ_]:=Sum[b[typ[[k]]-1,k],{k,Length[typ]}]\n");
printf("B[m_,␣n_]␣:=␣Sum[k!^2*StirlingS2[m+1,k+1]*StirlingS2[n+1,k+1],\n");
printf("␣␣␣{k,0,Min[m,n]}];\n");
```

This code is used in section 1.

6. ⟨ Report what we just did 6 ⟩ ≡

```
fprintf(stderr, "removing␣g%d:␣type␣(", m);
for (j = 0; j < l; j++) fprintf(stderr, "␣%d", x[j]);
fprintf(stderr, "␣)␣[");
for (j = 0; j < l; j++) fprintf(stderr, "␣b%d", xname[j]);
fprintf(stderr, "␣],␣n=%d,␣d=%d\n", p − 1, d);
printf("(*␣g%d␣*)␣", m);
if (l ≡ 0) printf("0+\n");
else {
  printf("B[%d,%d]", m − 1, n);
  for (j = 1; j < l; j++) printf("+b[%d,%d]B[%d,%d]", n, j, m − 1, n + 1 − j);
  printf("+brank[{");
  for (j = 0; j < l; j++) {
    if (j) printf(",");
    printf("%d", x[j]);
  }
  printf("}]B[%d,%d]+\n", m − 1, n + 1 − l);
}
```

This code is used in section 3.

7. Index.

⟨ Print the boilerplate to get Mathematica started 5 ⟩ Used in section 1.
⟨ Process the command line 2 ⟩ Used in section 1.
⟨ Remove girl m and reduce the parade 3 ⟩ Used in section 1.
⟨ Renumber the blocks if block t is going away 4 ⟩ Used in section 3.
⟨ Report what we just did 6 ⟩ Used in section 3.

RANK-PARADE2