1. Intro. Given a graph this program produces the unsatisfiable SAT problem in Tseytin's classic paper about lower bounds for regular resolution. The output is suitable for input to SAT0, SAT1, etc.

```
#include "gb_graph.h"   /* we use the GB_GRAPH data structures */
#include "gb_save.h"    /* and input the graph in the usual way */
  char bit[1000];

  int main(int argc, char *argv[])
  {
    register d, k, j;
    register Graph *g;
    register Vertex *u, *v;
    register Arc *a;

    ⟨Process the command line 2⟩;
    ⟨Generate the clauses 3⟩;
  }
```

2. ⟨Process the command line 2⟩ ≡

```
  if (argc ≠ 2) {
    fprintf(stderr, "Usage: %s foo.gb\n", argv[0]);
    exit(−1);
  }
  g = restore_graph(argv[1]);
  if (¬g) {
    fprintf(stderr, "I couldn't reconstruct graph %s!\n", argv[1]);
    exit(−2);
  }
  printf("~ sat-tseytin %s\n", argv[1]);
```

This code is used in section 1.

3. There's one variable in the SAT program for each edge of g. So we call that variable $u.v$, when the edge runs from u to v. For each vertex v we generate 2^{d-1} clauses, where d is the degree of v; each of those clauses involves the d variables adjacent to v. There's one clause for each way to complement an *odd* number of literals, except that we complement an *even* number when v is the very first vertex.

⟨Generate the clauses 3⟩ ≡

```
  for (v = g→vertices; v < g→vertices + g→n; v++) {
    for (d = −1, a = v→arcs; a; a = a→next) d++;
    while (1) {
      ⟨Generate a clause for the current bit setting 4⟩;
      for (k = 0; bit[k]; k++) bit[k] = 0;
      if (k ≡ d) break;
      bit[k] = 1;
    }
  }
```

This code is used in section 1.

4. ⟨ Generate a clause for the current *bit* setting 4 ⟩ ≡

```
for (j = (v > g→vertices), k = 0, a = v→arcs; a; a = a→next, j ⊕= bit[k], k++) {
  printf(" ");
  if (k ≡ d) ⟨ Adjust the parity of the final literal 5 ⟩
  else if (bit[k]) printf("~");
  u = a→tip;
  if (u < v) printf("%s.%s", u→name, v→name);
  else printf("%s.%s", v→name, u→name);
}
printf("\n");
```

This code is used in section 3.

5. ⟨ Adjust the parity of the final literal 5 ⟩ ≡

```
{
  if (j) printf("~");
}
```

This code is used in section 4.

6. Index.

SAT-TSEYTIN