1. **Intro.** This little program outputs clauses that are satisfiable if and only if the graph $g$ can be $c$-colored, given $g$ and $c$.

   (It generalizes SAT-PIGEONS, which is the case where $g = K_m$ and $c = n$.)

   Suppose the graph has $m$ edges and $n$ vertices. Then there are $nc$ variables $v.k$, meaning that vertex $v$ gets color $k$. And there are $n$ clauses of size $c$ (to ensure that each vertex gets at least one color), plus $mc$ clauses of size 2 (to ensure that adjacent vertices don’t share a color).

   ```c
   #include <stdio.h>
   #include <stdlib.h>
   #include "gb_graph.h"
   #include "gb_save.h"

   int c;
   main (int argc, char *argv[]) {
      register int i, j, k;
      register Arc *a;
      register Graph *g;
      register Vertex *v;
      ⟨Process the command line 2⟩;
      ⟨Generate the positive clauses 3⟩;
      ⟨Generate the negative clauses 4⟩;
   }

   2. ⟨Process the command line 2⟩
   if (argc ≠ 3 ∨ sscanf(argv[2], "%d", &c) ≠ 1) {
      fprintf(stderr, "Usage: %s foo.gb.%d
", argv[0]);
      exit(-1);
   }
   g = restore_graph(argv[1]);
   if (!g) {
      fprintf(stderr, "I couldn’t reconstruct graph’s!
", argv[1]);
      exit(-2);
   }
   if (c ≤ 0) {
      fprintf(stderr, "c must be positive!
");
      exit(-3);
   }
   printf("\n\sat-color,%d
", argv[1], c);

   This code is used in section 1.

   3. ⟨Generate the positive clauses 3⟩
   for ($v = g$-vertices; $v < g$-vertices + $g$-$n$; $v++$) {
      for ($k = 1; k ≤ c; k++$) printf("%d", v.name, k);
      printf("\n");
   }

   This code is used in section 1.
4. (Generate the negative clauses $4$) \( \equiv \)
   \begin{verbatim}
   for (k = 1; k <= c; k++)
     for (v = g-vertices; v < g-vertices + g-n; v++)
       for (a = v-arcs; a; a = a-next)
         if (a-tip > v) printf("~%s.%d \~%s.%d\n", v-name, k, a-tip-name, k);
   \end{verbatim}
This code is used in section 1.
5. Index.

a: 1.
Arc: 1.
arc: 4.
argv: 1, 2.
argv: 1, 2.
c: 1.
exit: 2.

fprintf: 2.
g: 1.
Graph: 1.
i: 1.
j: 1.
k: 1.
main: 1.
name: 3, 4.
next: 4.

printf: 2, 3, 4.
restore_graph: 2.
scanf: 2.
stderr: 2.
tip: 4.
v: 1.

Vertex: 1.
vertices: 3, 4.
(Generate the negative clauses 4) Used in section 1.
(Generate the positive clauses 3) Used in section 1.
(Process the command line 2) Used in section 1.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intro</td>
<td>1</td>
</tr>
<tr>
<td>Index</td>
<td>5</td>
</tr>
</tbody>
</table>