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 differs from SAT-COLOR-LOG2 because it uses a shorter way to compare binary labels.

Suppose the graph has $m$ edges and $n$ vertices, and let $t = \lceil \lg c \rceil$. Then there are $nt$ variables $v.k$, meaning that vertex $v$ gets color $(v.1 \ v.2 \ldots v.t)$. The final bit $v.t$ is sometimes irrelevant; for example, when $c = 3$, colors 10 and 11 are considered to be the same, we can consider the three possible colors to be 00, 01, and 1. When $c = 5$ the five possible colors are 000, 001, 01, 10, and 1.

There are $cm$ clauses of size $2t$ or $2t - 2$ 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, k, kk, t;
    register Arc *a;
    register Graph *g;
    register Vertex *u, *v;
    (Process the command line 2);
    for (t = 0; c > (1 << t); t++) ;
    for (v = g->vertices; v < g->vertices + g->n; v++)
        for (a = v->arcs; a; a = a->next) {
            u = a->tip;
            if (u < v) (Generate clauses to keep u and v from having the same color 3);
        }
}
```

2. (Process the command line 2) 
   if ($\text{argc} \neq 3 \lor \text{sscanf}(\text{argv}[2], "\%d", \&c) \neq 1$) {
       fprintf(stderr, "Usage: %s foo.gb c \n", argv[0]);
       exit(-1);
   }
   g = restore_graph(argv[1]);
   if (!g) {
       fprintf(stderr, "I couldn’t reconstruct %s\n", argv[1]);
       exit(-2);
   }
   if ($c \leq 0$) {
       fprintf(stderr, "c must be positive!\n"east;
       exit(-3);
   }
   printf("-sat-color-log3 %s\n", argv[1], c);
   This code is used in section 1.
3. \{ Generate clauses to keep \( u \) and \( v \) from having the same color \} \equiv
\[
\begin{align*}
&\text{for } (k = c; k < c + c; k++) \{ \\
&\quad \text{for } (i = t, \text{\( kk = k \); } i; i--) \{ \\
&\qquad \text{if } (i < t \lor k \geq (1 \ll t)) \{ \\
&\qquad \quad \text{printf ("%s%s.%d %s%s.%d", \( kk \& 1 ? \"~\" : \"\", u-name, i, \( kk \& 1 ? \"~\" : \"\", v-name, i); \\
&\qquad \quad \text{\( kk \gg= 1; \)}} \\
&\qquad \} \\
&\quad \text{printf ("\n")}; \\
&\}\}
\end{align*}
\]
This code is used in section 1.
4. Index.

\textbf{Arc}: 1.
\texttt{arc}:
\texttt{argc}:
\texttt{argv}:
\texttt{c}:
\texttt{exit}:
\texttt{fprintf}:
\texttt{Graph}:
\texttt{i}:
\texttt{k}:
\texttt{kk}:
\texttt{name}:
\texttt{next}:
\texttt{printf}:
\texttt{restore\_graph}:
\texttt{scanf}:
\texttt{stderr}:
\texttt{t}:
\texttt{tip}:
\texttt{u}:
\texttt{v}:
\textbf{Vertex}:
\texttt{vertices}:
(Generate clauses to keep $u$ and $v$ from having the same color 3) Used in section 1.
(Process the command line 2) Used in section 1.