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). Plus \( n \binom{c}{2} \) “exclusion clauses,” to ensure that no vertex gets more than one 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);
    (Generate the exclusion clauses 5);
}
```

2. (Process the command line 2) \( \equiv \)
```c
if (argc \neq 3 \lor sscanf(argv[2], "%d", &c) \neq 1) {
    fprintf(stderr, "Usage:\%s foo.gb c
    exit(-1);
}
```

```c
g = restore_graph(argv[1]);
if (!g) {
    fprintf(stderr, "I couldn’t reconstruct\%s!
    exit(-2);
}
if (c \leq 0) {
    fprintf(stderr, "c must be positive!
    exit(-3);

printf("%sat-color-exclusion\%s\d\n\n", argv[1], c);
```

This code is used in section 1.

3. (Generate the positive clauses 3) \( \equiv \)
```c
for (v = g-vertices; v < g-vertices + g-n; v++) {
    for (k = 1; k \leq c; k++) printf("%s.%d", v-name, k);

    printf("\n");
```

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

5. (Generate the exclusion clauses 5) \( \equiv \)
   \[ j = 1; \ j \leq c; \ j++ \]
   \[ k = j + 1; \ k \leq c; \ k++ \]
   \[ (v = g\text{-vertices}; \ v < g\text{-vertices} + g\text{-n}; \ v++) \ \text{printf } ("\%s . \%d . \%d \n", v\text{-name}, j, v\text{-name}, k); \]
   This code is used in section 1.
6. Index.

a: 1.
 Arc: 1.
 arcs: 4.
 argc: 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, 5.
 next: 4.
 printf: 2, 3, 4, 5.
 restore_graph: 2.
 scanf: 2.
 stderr: 2.
 tip: 4.
 v: 1.
 Vertex: 1.
 vertices: 3, 4, 5.
(Generate the exclusion clauses 5) Used in section 1.
(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.