1. Intro. A trivial program to create an SGB graph. The first line of standard input lists the vertex names; the remaining lines list the (directed) edges, as triples `x y` *d*.

An optional command-line argument gives the name of the graph. For example, if the name is `test`, the graph is saved as `/tmp/test.gb`.

```
#define maxn 100000    /* at most this many vertices */
#define maxl 3    /* maximum length of vertex name */
#define bufsize (maxl + 1) * maxn + 2

#include <stdio.h>
#include <stdlib.h>
#include "gb_graph.h"
#include "gb_save.h"
  char buf[bufsize + 1];
  char names[maxn][maxl + 1];
  char nbuf[maxl + 1];
  char filenamebuf[ID_FIELD_SIZE + 8] = "/tmp/makegraph.gb";
  int main(int argc, char *argv[])
  {
    register int j, k, m, n, s;
    register long d;
    Graph *g;
    Vertex *u, *v;
    ⟨ Input the vertices 2 ⟩;
    ⟨ Input the edges 3 ⟩;
    ⟨ Output the graph 4 ⟩;
  }
```

2. ⟨ Input the vertices 2 ⟩ ≡

```
  buf[bufsize] = '\n';
  if (¬fgets(buf, bufsize, stdin)) {
    fprintf(stderr, "Couldn't␣read␣the␣variable-name␣line!\n");
    exit(−1);
  }
  for (n = k = 0; n < maxn; n++) {
    while (buf[k] ≡ '␣') k++;
    if (buf[k] ≡ '\n') break;
    for (j = 0; buf[k] ≠ '␣' ∧ buf[k] ≠ '\n' ∧ j ≤ maxl; j++, k++) names[n][j] = buf[k];
    if (j > maxl) {
      fprintf(stderr, "Vertex␣name␣is␣too␣long!␣%s", buf − k − j);
      exit(−2);
    }
  }
  g = gb_new_graph(n);
  for (k = 0; k < n; k++) (g→vertices + k)→name = gb_save_string(names[k]);
  hash_setup(g);
  printf("I've␣created␣a␣graph␣with␣%d␣vertices...\n", n);
```

This code is used in section 1.

3. ⟨ Input the edges 3 ⟩ ≡

```
for ($m = 0$; ; $m$++) {
  if ($\neg fgets(buf, bufsize, stdin)$) break;
  for ($k = 0$; $buf[k] \equiv$ '␣'; $k$++) ;
  for ($j = 0$; $buf[k] \neq$ '␣' $\land j < maxl$; $j$++, $k$++) $nbuf[j] = buf[k]$;
  $nbuf[j] =$ '\0';
  $u = hash\_out(nbuf)$;
  if ($\neg u$) {
    $fprintf(stderr,$ "Unknown␣first␣vertex:␣%s", $buf)$;
    $exit(-3)$;
  }
  for ( ; $buf[k] \equiv$ '␣'; $k$++) ;
  for ($j = 0$; $buf[k] \neq$ '␣' $\land j < maxl$; $j$++, $k$++) $nbuf[j] = buf[k]$;
  $nbuf[j] =$ '\0';
  $v = hash\_out(nbuf)$;
  if ($\neg v$) {
    $fprintf(stderr,$ "Unknown␣second␣vertex:␣%s", $buf)$;
    $exit(-4)$;
  }
  for ( ; $buf[k] \equiv$ '␣'; $k$++) ;
  if ($buf[k] \equiv$ '-') $s = -1, k$++; else $s = +1$;
  for ($d = 0$; $buf[k] \geq$ '0' $\land buf[k] \leq$ '9'; $k$++) $d = 10 * d + buf[k] -$ '0';
  $gb\_new\_arc(u, v, s * d)$;
}
$printf($"␣and␣%d␣arcs...\n", $m)$;
```

This code is used in section 1.

4. ⟨ Output the graph 4 ⟩ ≡

```
if ($argc > 1$) {
  $sprintf(g\rightarrow id,$ "%.*s", ID_FIELD_SIZE $- 1, argv[1])$;
  $sprintf(filenamebuf,$ "/tmp/%.*s.gb", ID_FIELD_SIZE $- 1, argv[1])$;
}
$save\_graph(g, filenamebuf)$;
$printf($"␣and␣file␣%s␣holds␣the␣result.\n", $filenamebuf)$;
```

This code is used in section 1.

5. Index.

MAKEDIGRAPH