1.   Intro.   We output an Eulerian trail of the (undirected) graph named on the command line. (Each edge is considered to be two directed arcs; thus it is traversed in both directions.)

   If the graph isn't connected, we consider only the vertices that are reachable from the first one, $g\rightarrow vertices$.

```
#include <stdio.h>
#include <stdlib.h>
#include "gb_graph.h"
#include "gb_save.h"
```
   ⟨ Subroutine 3 ⟩
   $main(\textbf{int } argc, \textbf{char } *argv[\,])$
   {
      register int k;
      Graph $*g$;
      Vertex $*u$, $*v$;
      Arc $*a$;
      ⟨ Input the graph 2 ⟩;
      ⟨ Traverse depth first 4 ⟩;
      ⟨ Output the trail 5 ⟩;
      $printf(\texttt{"\textbackslash n"})$;
   }

2.   ⟨ Input the graph 2 ⟩ ≡
   if $(argc \neq 2 \vee \neg(g = restore\_graph(argv[1])))$ {
      $fprintf(stderr, \texttt{"Usage:\_\%s\_foo.gb\textbackslash n"}, argv[0])$;
      $exit(-1)$;
   }
   $fprintf(stderr, \texttt{"OK,\_I've\_input\_'\%s'.\textbackslash n"}, argv[1])$;
   $gb\_new\_edge(g\rightarrow vertices, g\rightarrow vertices + g\rightarrow n, 0)$;   /∗ dummy edge ∗/

This code is used in section 1.

3.   Subroutine $dfs(u,v)$ sets $v\rightarrow parent = u$ and $v\rightarrow nav$ to the vertex that follows u in v's adjacency list. It also explores all vertices reachable from v that haven't already been seen.

```
#define parent  v.V
#define nav  w.A
```
⟨ Subroutine 3 ⟩ ≡
   void $dfs(\textbf{register Vertex } *u, \textbf{register Vertex } *v)$
   {
      register Vertex $*w$;
      register Arc $*a$;

      $v\rightarrow parent = u$;
      for $(a = v\rightarrow arcs;\ a;\ a = a\rightarrow next)$ {
         $w = a\rightarrow tip$;
         if $(w \equiv u)$ $v\rightarrow nav = a\rightarrow next$;
         else if $(w\rightarrow parent \equiv \Lambda)$ $dfs(v,w)$;
      }
   }

This code is used in section 1.

4.   ⟨ Traverse depth first 4 ⟩ ≡
   $dfs(g\rightarrow vertices + g\rightarrow n, g\rightarrow vertices)$;

This code is used in section 1.

5.   Now the Eulerian traversal is beautifully simple.

⟨ Output the trail 5 ⟩ ≡
   for (*v* = *g*→*vertices*; *v* ≠ *g*→*vertices* + *g*→*n*;) {
     printf ("␣%s", *v*→*name*);
     a = *v*→*nav*;
     if (¬*a*) *a* = *v*→*arcs*;
     v→*nav* = *a*→*next*;
     v = *a*→*tip*;
   }

This code is used in section 1.

6.   Index.

⟨ Input the graph 2 ⟩   Used in section 1.
⟨ Output the trail 5 ⟩   Used in section 1.
⟨ Subroutine 3 ⟩   Used in section 1.
⟨ Traverse depth first 4 ⟩   Used in section 1.

EULER-TRAIL