## Testing protocol for Galant.

1. Create a new graph with two nodes and one edge; give both nodes and the edge a color, weight, and label; add another node and edge using ^n and ^e, respectively; test a variety of keyboard shortcuts such as ones to delete nodes and edges. The relevant ones are:
   * ^L, ^l – toggle edge/node label visibility
   * ^W,^w – toggle edge/node weight visibility
   * DEL+e. DEL+n – delete edge/node (must hold down DEL while pressing e or n)
   * ^i – smart reposition
2. Add more nodes and edges, do some deletions, and try the undo (^z) and redo (^y) options.
3. Open test (in Example\_Graphs) and move some nodes around – can skip this if did node movement in step 0.
4. Run bfs on unweighted\_8 (in Example\_Graphs); in the middle of running the algorithm, move a few nodes around; these changes should persist during backward and forward motion as well as after the algorithm is done. Run the algorithm multiple times with directedness turned off and on.
5. Run dfs\_d on unweighted\_8 (in Example\_Graphs), need node and edge labels, no weights, graph must be directed. Run the algorithm multiple times. *Be sure to designate the graph as directed first.*
6. Run dijkstra on weighted\_6 (in Example\_Graphs), with graph both undirected and directed; in undirected case, node 3 is distance 2 from the start via path 0-2-3; in the directed case it’s distance 3 via the edge 0->3; need weights on both edges and nodes
7. Run kruskal on weighted\_6, need edge weights only; there’s text commentary
8. Run insertion\_sort on the graph in Example\_Graphs/0-SortingGraphs called medium\_sorting\_test, need node weights only. Any attempt to move a node should get no reaction.
9. Run binary\_tree on an empty graph, no weights or labels, try exporting an intermediate state; also try moving nodes around
10. Load crossing\_test (in Example\_Graphs) and make sure it draws properly (responds to changes in window size). Try both contiguous and non-contiguous positionInLayer attributes. Contiguous positions lead to evenly spaced placement.
11. Write and execute some algorithms that make full use of macros, functions, etc. To be spelled out more explicitly later.

### Crossing Algorithms

1. run both barycenter and mce (in Research/Layered-Graphs/Algorithms) on crossing\_test (in Layered-Graphs/Graphs), need node weights for barycenter
2. ditto for r\_100\_110\_10\_0\_0p7 (in Layered-Graphs/Graphs)
3. run sifting on crossing\_test
4. run mce on crossing\_test
5. run all three on r\_100\_110\_10\_0\_0p7