## 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
2. Open test (in Example\_Graphs) and move some nodes around
3. Run bfs on eight\_node\_graph (in Example\_Graphs); in the middle of running the algorithm, change visibilities of weights and labels and move a few nodes around; these changes should persist during backward and forward motion as well as after the algorithm is done.
4. Run dfs\_d on eight\_node\_graph (in Example\_Graphs), need node and edge labels, no weights, graph must be directed
5. Run dijkstra on weighted\_example (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
6. Run kruskal on weighted\_example, need edge weights only; there’s text commentary
7. Run insertion\_sort on sorting\_test (in Example\_Graphs), need node weights only
8. Run binary\_tree on an empty graph, no weights or labels, try exporting an intermediate state; also try moving nodes around
9. Load 1\_test (in Crossing\_Graphs) and make sure it draws properly (responds to changes in window size)
10. Write and execute some algorithms that make full use of macros, functions, etc. To be spelled out more explicitly later.

### Crossing Algorithms (Skip for now)

1. run both barycenter and mce (in Crossing-Algorithms) on 1\_test (in Crossing Graphs), need node weights for barycenter
2. ditto for r\_100\_110\_10\_0\_0p7 (in Crossing\_Graphs)
3. run sifting on 1\_test