```
[q]Minimum: 30[/q]
[q]Core: 35[/q]
[q]Completion: 20[/q]
[q]Challenge: 15[/q]

Minimum out of 30:
* [15] The program reads the data and draws a map.
* [15] It constructs a graph structure using collections of intersections, segments, and roads.

Core out of 35 (up to 65):
* [15] The map can be zoomed and panned, and the user can select intersections with the mouse,
and some details are shown.
* [10] Road names can be entered into the text box and all segments are highlighted on the map.
Can be either exact or prefix matches, check that all segments highlighted.
* [5] The names of all roads joining the selected intersection are outputted, without duplicates.
* [5] The graph (not exhaustive search) is used to find the roads joining a selected
intersection.
Getting from node to segment to road should be constant time.

Completion out of 20 (up to 85):
* [15] Correct code for a trie structure, with methods to add an element and find all elements
with a given prefix.
* [5] Trie is used to output the names of all roads which match a prefix in the search box, as
well
as highlighting them on the map. If the prefix exactly matches a road name, only output and
highlight road(s) of that name.

Challenge out of 15 (up to 100 max, with 5 spare marks here):
* [5] A quad-tree structure is used to quickly find intersections near a mouse click.
* [5] The quad-tree retrieves the closest intersection in all cases.
* [5] The polygon data is used to draw a nice map with lakes, parks, airports, a coastline, etc.
* [5] The GUI is improved, with one of mouse-based panning and zooming or adding a drop-down
suggestion box to the search bar.

Final mark calculation:

[q]Minimum: 30[/q] + [q]Core: 35[/q] + [q]Completion: 20[/q] + [q]Challenge: 15[/q]
```