Programming Lab:
Implement a version of A-Star in Prolog.   POINTS: 23

A-Star is a version of Shortest Path that also adds a heuristic estimate to the path distance to goal. The graph will be stored as edges; each edge will be stored (twice) in both directions.

```
edge(va, vb, 40).
edge(vb, va, 40).
edge(va, vh, 40).
edge(vh, va, 40).
```

The X, Y, coordinate of each vertex is also stored.

```
coordinate(va, 90, 160).
coordinate(vb, 70, 140).
```

A document discussing my own solution also gives you the code to compute the SLD (Straight Line Distance) which serves as the Heuristic. In my code I have a variable PCost (the cost of the partial solution), and a variable PHCost (which is the PCost + SLD = PHCost to the Goal vertex).
P for Partial, H for Heuristic.

You must use the following state configuration:

[PCost, PHCost, CurrentV, vi… vj, vstart]

And when you reach the goal you will have:

[PCost, PHCost, vgoal, vi, … vj, vstart]

I refer to the set of configurations as a “queue” but I realize while coding that we don’t necessarily need a queue (you can use a queue, and sort it if you want, but you don’t have to).
What we do need at each “step” is the current minimal PHCost configuration.
So my “queue” is really just a list of configurations.

Here is a sample list of configuration for a specific run:

[[40,130.55,vb,va],[40,93.85,vh,va],[90,121.62,vi,va],[150,280,vc,va]]

You can tell the start state was “va” and the neighbors of “va” are [vb, vh, vi, vc]. The minimal PHCost is found here: [40,93.85,vh,va],

Here is the next list of configurations for this specific run. You can see that vertex “vh” has been expanded with (new) neighbors [vg, vp, vi].

[[60,110,vg,vh,va],[160,200,vp,vh,va],[120,151.62,vi,vh,va],[40,130.55,vb,va],[90,121.62,vi,va],[150,280,vc,va]]

Use the following helper function (rule) to hide details:

callastar(Vstart, Vgoal, Path) where path is a configuration such as: [60,60,vg,vh,va] given that “va” is the start state and “vg” is the goal state. When you reach the Goal the SLD=0 and the PCost and PHCost are the same.

Thus, callastar(Vstart, Vgoal, Path) will call the A-Star code and initial the configuration at the start state:

```
callastar(Start, Goal, Path) :-

       distanceSL(Start, Goal, SLD),

       astar([[0,SLD,Start]], Goal, [], Path).
```

In my code, callastar(Vstart, Vgoal, Path) also checks for trivial solutions (such as Vstart = Vgoal, or Vgoal is a neighbor of Vstart).

Just as a reference, my code is 85 lines (including blank lines) and uses 17 rules. It could probably be tighter. For example, a more complex rule might replace two simpler rules. But then the code is also more complex.
For this lab you will also find two files. One provides an example problem and Prolog code for computing straight line distance. Another file explains how I coded the problem.