1. Describe the heuristic you designed, and show that it is admissible.
My Heuristic function is a combination of Manhattan distance and Euclidean distance in the following form.

if **M = Manhattandistance(x,y) =** $|goalnode.x - x1| + |goalnode.y - y1|$

E = Euclideandistance(x,y) = $\sqrt{(goalnode.x - x1)2 + (goalnode.y - y1)2}$

then, $h(x,y) = \sqrt{(M*M)/4 + (E*E)}$

I got very optimal results with this heuristic function.
It resulted not only in optimal path when compared to Manhattan, but also explored fewer nodes when constant g(n) was used and explored a few more nodes when bottom-favouring g(n) cost function was used.

Proof of admissibility:
A heuristic function is deemed admissible if it never overestimates the distance of the current node to the goal node when compared to the optimal path. Since in our problem, we cannot traverse diagonally along the grid, it makes Manhattan distance admissible.
Since Manhattan is admission, all I need to prove is that my heuristic function is less than or equal to Manhattans heuristic function value.

for eg: if current node is (1,1) and goal node is (3,3)
Manhattan heuristics will give result as 4
while my heuristic function will give result as $\sqrt{12}$ ~= 3.4 which is less than 4 and hence admissible.

Mathematically the above h(x,y) can be proved to be less than Manhattan distance and hence the below equation.

$$\sqrt{(M*M)/4 + (E*E)} \leq |goalnode.x - x1| + |goalnode.y - y1|$$

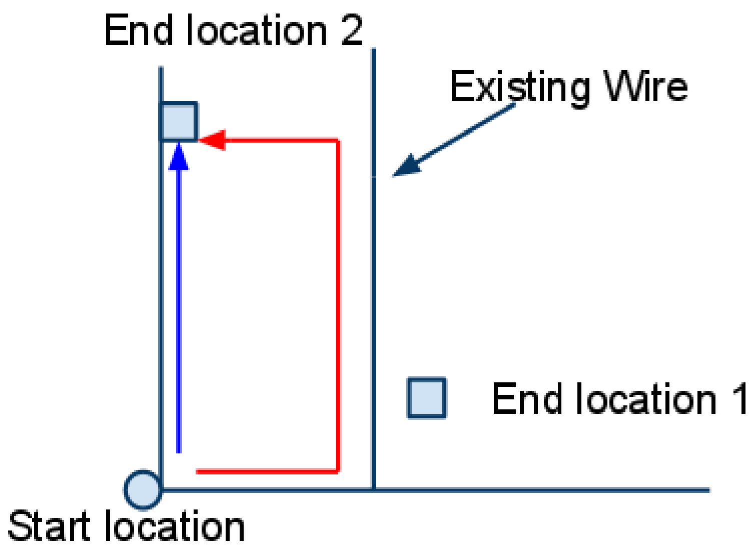
2. Suppose you were instead given several end locations, and your task was to place a wire that began at the start location, and ended at any one of the end locations the algorithm prefers. Note that all, none,or a subset of the end locations may be reachable. One obvious extension of the Manhattan-distance heuristic would be the Manhattan distance from the new wire's current ending location to the closest end location:

$$h((x, y)) = \min_{(x`,y`)} |x - x`| + |y - y`|$$

a. Is this heuristic admissible?
b. When might it fail (lead to many nodes being expanded)?
c. Provide another heuristic for the new problem, and argue that it is better than the

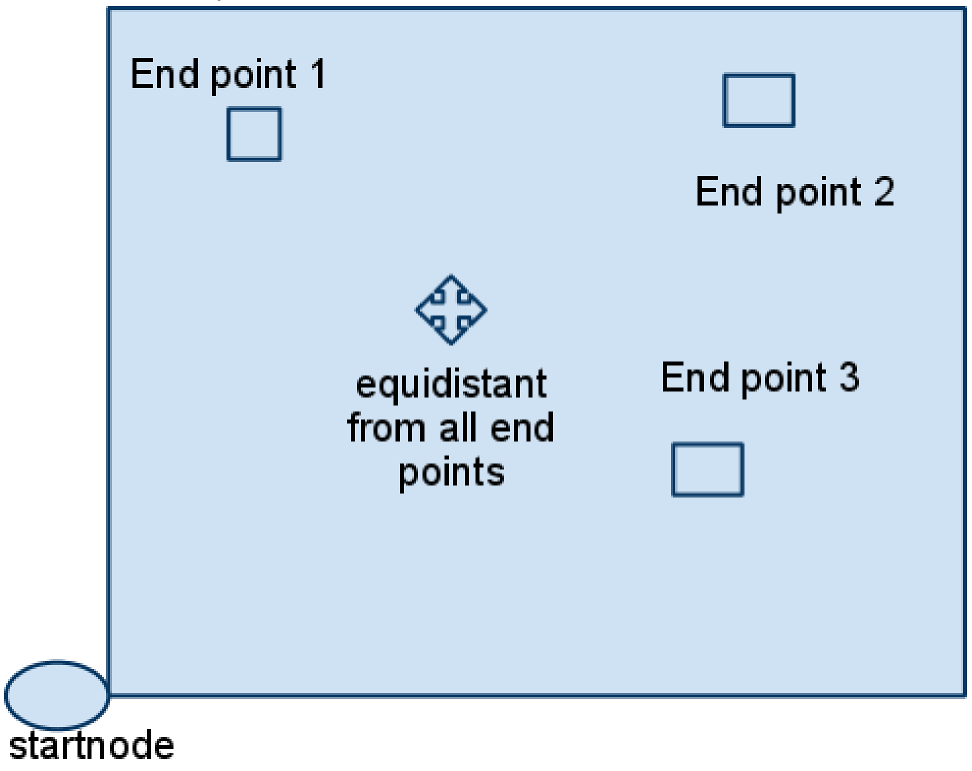
Manhattan
distance to closest goal heuristic.

Answer 2:
a> No this heuristic is not admissible because if we have a situation as shown below then by the above heuristic we will start to move towards end location 1 since its closest end point but we will hit the already existing wire and start moving upwards exploring children along the wire, and now at some point we are closer to end location 2 and we start moving towards it. The red line shows the path taken which is longer than the path that could have ended up to endlocation 2 represented by blue arrow.
So just because the heuristic function thought that end location 1 is closest and started moving towards it and hit a road block at the end and started moving towards next closest endlocation will result in an overestimation of the cost. and hence not admissible.



b> as explained in part a> it fails mainly because, if when it starts moving towards a end location and realizes that there is no clear path to end location, and starts moving towards another end location, we end up expanding more nodes since we take longer path in the fail case as explained.
So basically the heuristic fails if the there is one end location closest to the start node but is unreachable because of wires surrounding it completely, But since manhattan distance doesnt know about it, it starts moving towards the node (which is apparently unreachable) and finally when it realizes its unreachable(since its cant explore nodes around the unreachable node) it starts moving towards another end location making the path longer and hence inadmissible.

c> First We find the point E on the grid that is equidistant (EUCLIDEAN) from all end points. Now in the heuristic function we chose the child nodes which is closest to this equidistant point E.(by euclidean distance) . Once we move towards that point we can randomly reach any of the end point and it can be proved that this will still be admissible (even if the chosen end point turns out to be unreachable making the algorithm take path towards another end point) and lesser than or equal to manhattan heuristics.



3. How did you test your code?
Do not simply enumerate test cases, but describe how you went about testing for specific algorithmic issues.

Answer 3.

TRACING THE TRAVERSAL - **Creating a visual grid with“visited” information**

One issue during this part of the assignment is the the difficulty in understanding how the traversal takes place.
I have a “visited” field in the GRIDNODE’s ,
So I created a visual grid and would display it with information about if a node in the grid was visited or not and print the output of the whole grid after every node was expanded

```
 ******************** 0 is not visited 1 is visited*****************************
  0   0   0   0   0   0   0   0   1   0
  0   0   0   0   1   1   1   1   1   1
  0  1  1  1  1  1  1  1  1  1
  0  1  1  1  1  0  1  1  1  0
  0  1  1  1  1  0  0  0  0  0
  0  1  1  1  1  0  0  0  0  0
  0  1  1  1  1  0  0  0  0  0
  0  1  0  0  0  0  0  0  0  0
  1  1  1  0  0  0  0  0  0  0
  0  1  0  0  0  0  0  0  0  0
*************************************************
  *  *  *  *  *  *  *  *  *  *
  N  N  N  N  *  *  *  *  P  E
  N  *  *  *  P  P  P  P  P  *
  N  *  *  *  P  N  *  *  *  *
  N  *  *  *  P  N  *  *  *  *
  N  *  *  *  P  N  *  *  N  *
  N  P  P  P  P  N  *  *  N  *
  N  S  N  N  N  N  *  *  N  *
  *  *  *  *  *  *  *  *  N  *
  *  *  *  *  *  *  *  *  N  *
```

That way it is easy to confirm if the algorithm is acting as per the desired behaviour.

Mapping 1-D struct to 2D grid(x,y)
I was using a one dimensional structure with Node information like, x coordinate, y coordinate, nodevisited or not, cost etc. and this structure has to be mapped to the two dimensional information of the grid, so that the information being accessed actually what is meant on the real grid. Meaning, the i value in struct[i] has to be mapped correctly to x,y of the grid.
So I used a 2 dimensional GRIDMATRIX[x][y] the value of which maps to the i value of the struct Node containing x and y as data. So at any point of time, GRIDMATRIX acts as an indexing matrix for the GRID STRUCT.

For Astar algorithm and UCS, one problem was creating/updating a cost matrix.
This could be solved using a parentindex or parentNode information for every Node.
so the cost of every node would be its cost + parent cost.

How would one test the final created path
This could be solved by backtracking from goal node to start node by getting current nodes parent and going back all the way to start node and push it along the stack while you do that, and pop them along and mark it along the visual grid to visualize the placement.