CS 133b **February 23, 2024**

Problem Set 7

Problem 1 (Forward Simulation) - 24 points:

part a

As described in class, the dimensions of the ending state will be the following:

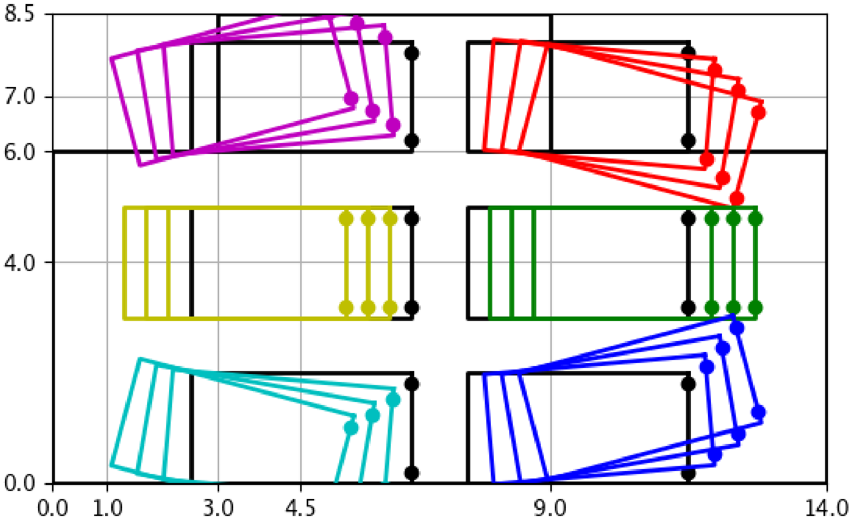
$$x_B = x_A + d \cdot cos\left(\theta_A + \frac{\Delta\theta}{2}\right) \cdot sinc\left(\frac{\Delta\theta}{2}\right)$$

$$y_B = y_A + d \cdot sin\left(\theta_A + \frac{\Delta\theta}{2}\right) \cdot sinc\left(\frac{\Delta\theta}{2}\right)$$

$$\theta_B = \theta_A + \Delta\theta$$

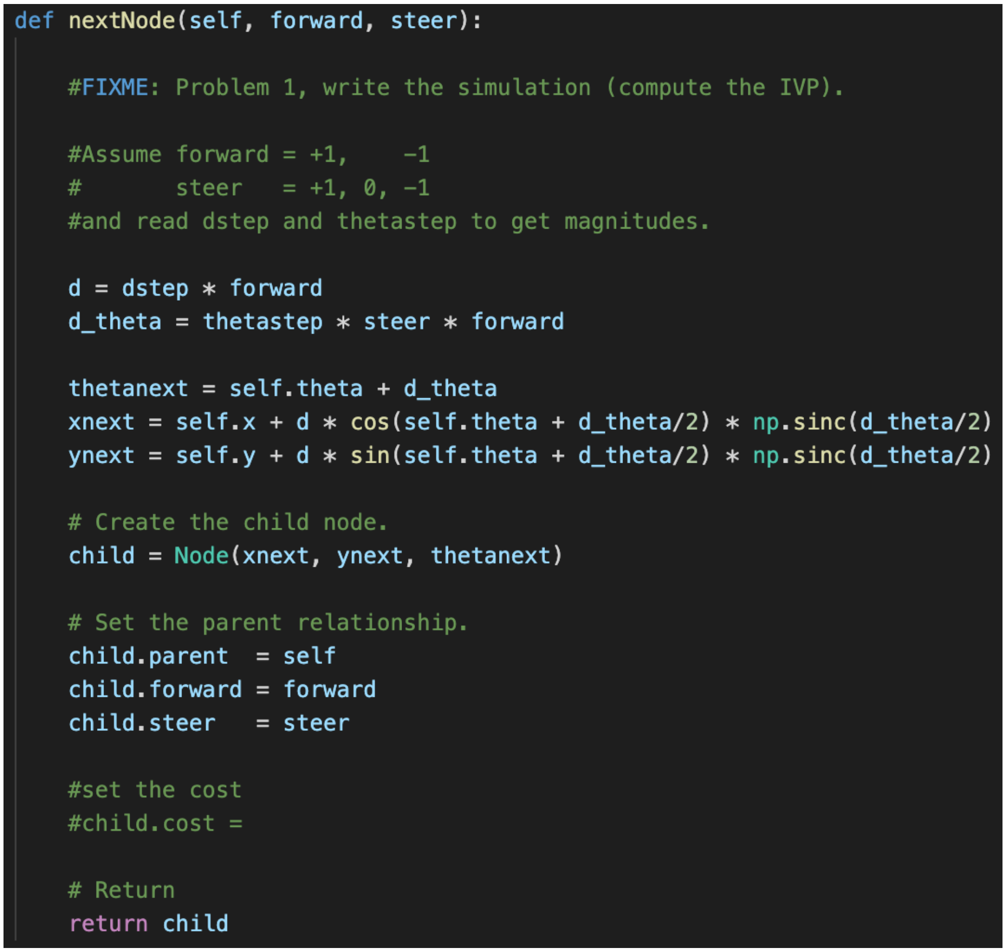
part b

Below is the output of the testdriving(). The code is on the next page.



Code for 1b

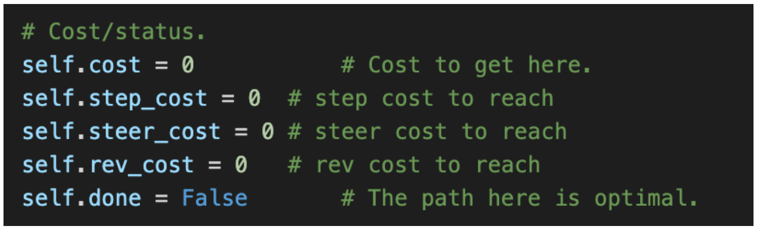
Below is the code written for 1b.



```python
def nextNode(self, forward, steer):

    #FIXME: Problem 1, write the simulation (compute the IVP).

    #Assume forward = +1,    -1
    #       steer   = +1, 0, -1
    #and read dstep and thetastep to get magnitudes.

    d = dstep * forward
    d_theta = thetastep * steer * forward

    thetanext = self.theta + d_theta
    xnext = self.x + d * cos(self.theta + d_theta/2) * np.sinc(d_theta/2)
    ynext = self.y + d * sin(self.theta + d_theta/2) * np.sinc(d_theta/2)

    # Create the child node.
    child = Node(xnext, ynext, thetanext)

    # Set the parent relationship.
    child.parent  = self
    child.forward = forward
    child.steer   = steer

    #set the cost
    #child.cost =

    # Return
    return child
```

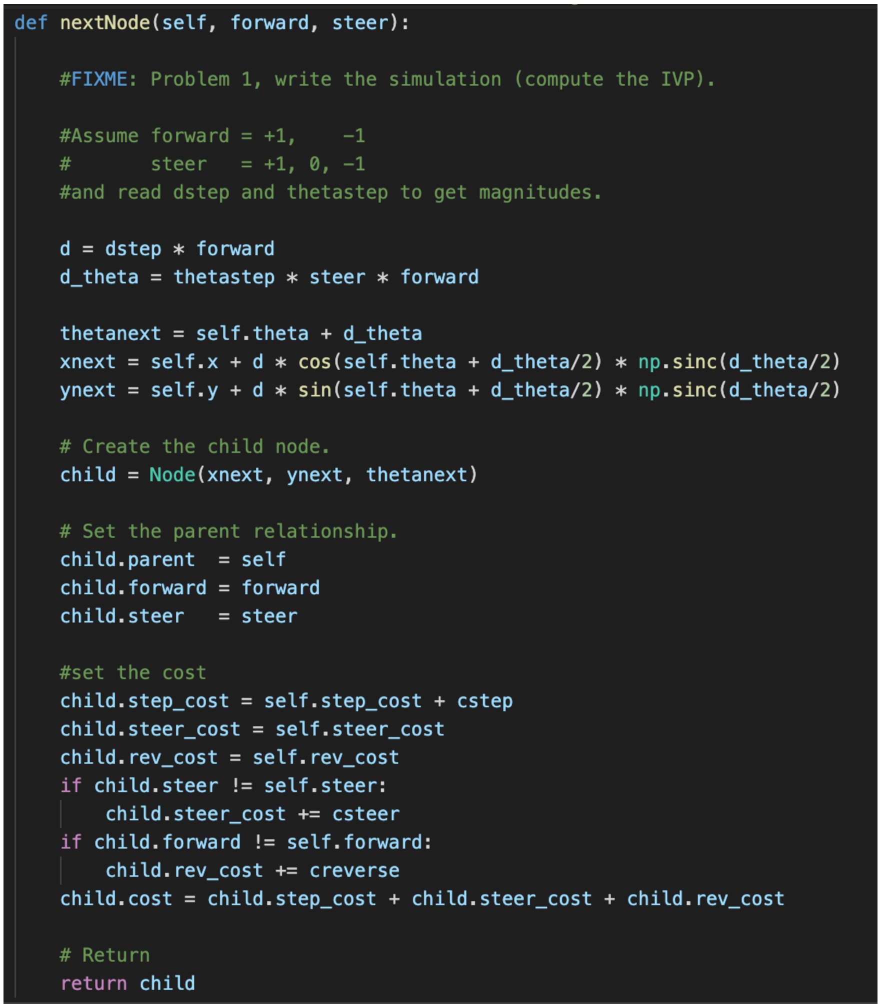
Problem 2 (Cost To Reach) - 16 points:

The init for the Node class was edited as shown below (three new attributes were added and set to zero). The attributes are step_cost, steer_cost, and rev_cost:



```
# Cost/status.
self.cost = 0           # Cost to get here.
self.step_cost = 0  # step cost to reach
self.steer_cost = 0 # steer cost to reach
self.rev_cost = 0   # rev cost to reach
self.done = False       # The path here is optimal.
```

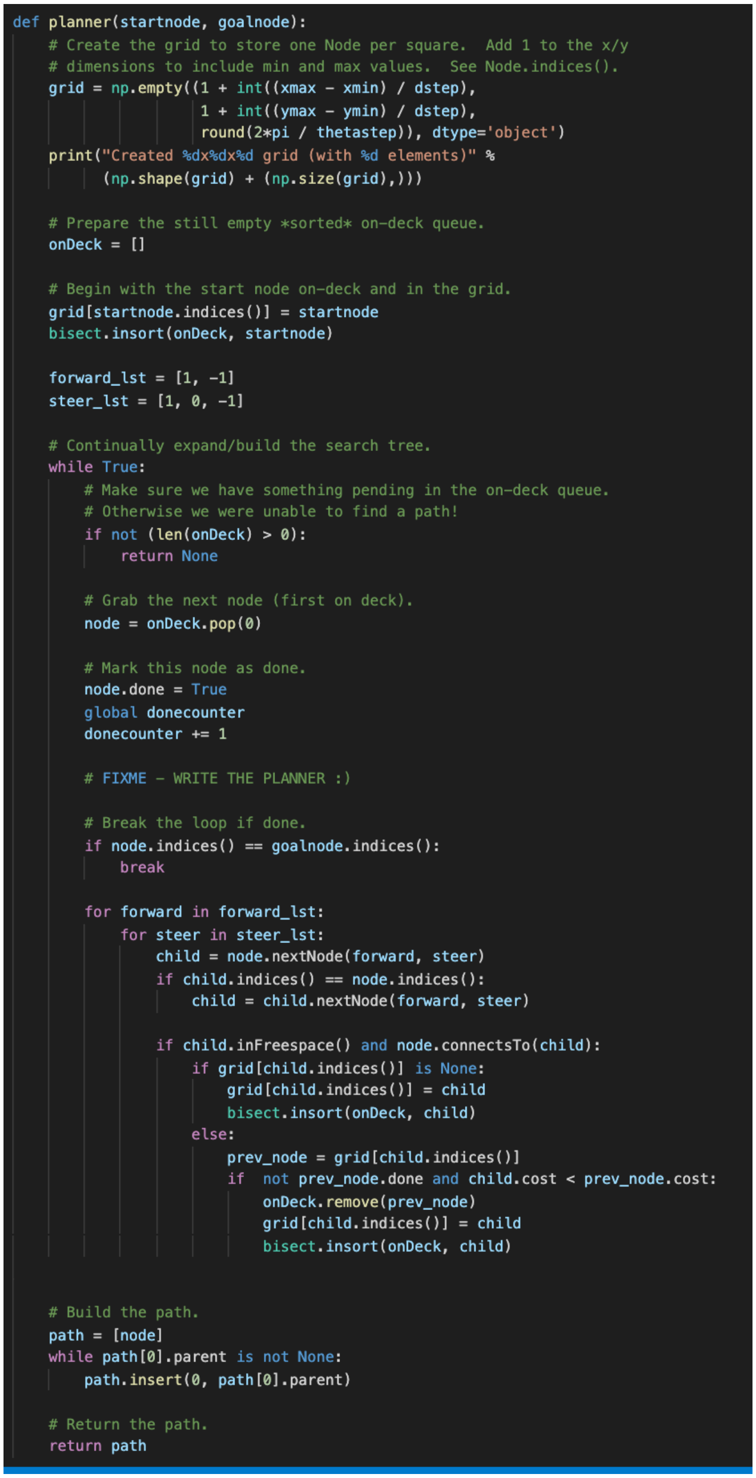
Updated NextNode() code is below. The output of testcosts() is the same as the correct output shown in the problem set.



```
def nextNode(self, forward, steer):

    #FIXME: Problem 1, write the simulation (compute the IVP).

    #Assume forward = +1,    -1
    #       steer   = +1, 0, -1
    #and read dstep and thetastep to get magnitudes.

    d = dstep * forward
    d_theta = thetastep * steer * forward

    thetanext = self.theta + d_theta
    xnext = self.x + d * cos(self.theta + d_theta/2) * np.sinc(d_theta/2)
    ynext = self.y + d * sin(self.theta + d_theta/2) * np.sinc(d_theta/2)

    # Create the child node.
    child = Node(xnext, ynext, thetanext)

    # Set the parent relationship.
    child.parent  = self
    child.forward = forward
    child.steer   = steer

    #set the cost
    child.step_cost = self.step_cost + cstep
    child.steer_cost = self.steer_cost
    child.rev_cost = self.rev_cost
    if child.steer != self.steer:
        child.steer_cost += csteer
    if child.forward != self.forward:
        child.rev_cost += creverse
    child.cost = child.step_cost + child.steer_cost + child.rev_cost

    # Return
    return child
```

Problem 3 (Planner Code) - 24 points:

Code for the planner function:



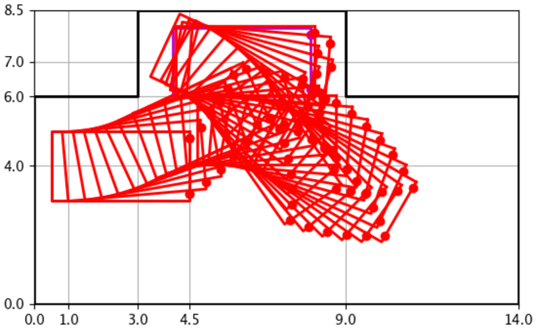
```python
def planner(startnode, goalnode):
    # Create the grid to store one Node per square.  Add 1 to the x/y
    # dimensions to include min and max values.  See Node.indices().
    grid = np.empty((1 + int((xmax - xmin) / dstep),
                     1 + int((ymax - ymin) / dstep),
                     round(2*pi / thetastep)), dtype='object')
    print("Created %dx%dx%d grid (with %d elements)" %
          (np.shape(grid) + (np.size(grid),)))

    # Prepare the still empty *sorted* on-deck queue.
    onDeck = []

    # Begin with the start node on-deck and in the grid.
    grid[startnode.indices()] = startnode
    bisect.insort(onDeck, startnode)

    forward_lst = [1, -1]
    steer_lst = [1, 0, -1]

    # Continually expand/build the search tree.
    while True:
        # Make sure we have something pending in the on-deck queue.
        # Otherwise we were unable to find a path!
        if not (len(onDeck) > 0):
            return None

        # Grab the next node (first on deck).
        node = onDeck.pop(0)

        # Mark this node as done.
        node.done = True
        global donecounter
        donecounter += 1

        # FIXME - WRITE THE PLANNER :)

        # Break the loop if done.
        if node.indices() == goalnode.indices():
            break

        for forward in forward_lst:
            for steer in steer_lst:
                child = node.nextNode(forward, steer)
                if child.indices() == node.indices():
                    child = child.nextNode(forward, steer)

                if child.inFreespace() and node.connectsTo(child):
                    if grid[child.indices()] is None:
                        grid[child.indices()] = child
                        bisect.insort(onDeck, child)
                    else:
                        prev_node = grid[child.indices()]
                        if  not prev_node.done and child.cost < prev_node.cost:
                            onDeck.remove(prev_node)
                            grid[child.indices()] = child
                            bisect.insort(onDeck, child)


    # Build the path.
    path = [node]
    while path[0].parent is not None:
        path.insert(0, path[0].parent)

    # Return the path.
    return path
```

Problem 4 (Parallel Parking) - 16 points:

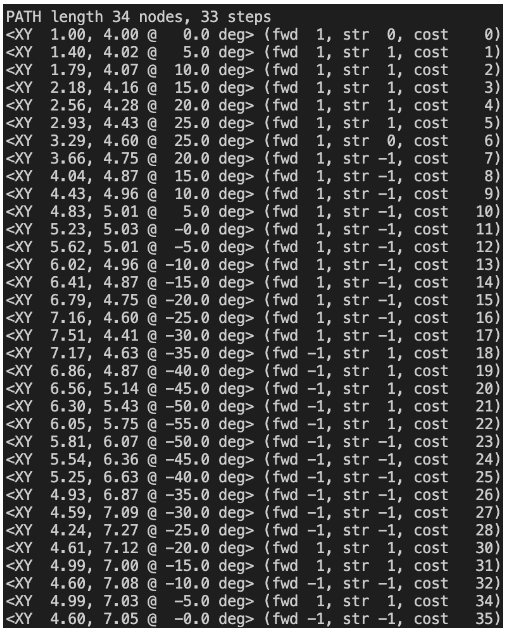
Using the following cost function:

$$cost = c_{step}N_{step} + c_{reverse}N_{reversals}$$

with $c_{step} = 1.0$ and $c_{reverse} = 0.5$, I got the following result:



A path was found with 33 steps. If the cost function included cost for changing steering then the number of steps became greater. Furthermore, without some cost for changing direction of movement (that is if $cost = c_{step}N_{step}$) then the number of steps also increased. Lastly, if $c_{reverse}$ became greater than c_{step}, then the number of steps also increased. Therefore, the cost was set as described above. Below is a screenshot of the cost values:



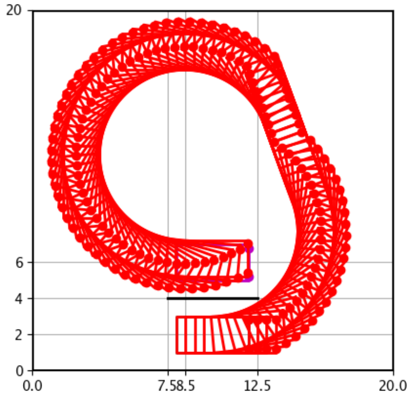
```
PATH length 34 nodes, 33 steps
<XY  1.00, 4.00 @   0.0 deg> (fwd  1, str  0, cost    0)
<XY  1.40, 4.02 @   5.0 deg> (fwd  1, str  1, cost    1)
<XY  1.79, 4.07 @  10.0 deg> (fwd  1, str  1, cost    2)
<XY  2.18, 4.16 @  15.0 deg> (fwd  1, str  1, cost    3)
<XY  2.56, 4.28 @  20.0 deg> (fwd  1, str  1, cost    4)
<XY  2.93, 4.43 @  25.0 deg> (fwd  1, str  1, cost    5)
<XY  3.29, 4.60 @  25.0 deg> (fwd  1, str  0, cost    6)
<XY  3.66, 4.75 @  20.0 deg> (fwd  1, str -1, cost    7)
<XY  4.04, 4.87 @  15.0 deg> (fwd  1, str -1, cost    8)
<XY  4.43, 4.96 @  10.0 deg> (fwd  1, str -1, cost    9)
<XY  4.83, 5.01 @   5.0 deg> (fwd  1, str -1, cost   10)
<XY  5.23, 5.03 @  -0.0 deg> (fwd  1, str -1, cost   11)
<XY  5.62, 5.01 @  -5.0 deg> (fwd  1, str -1, cost   12)
<XY  6.02, 4.96 @ -10.0 deg> (fwd  1, str -1, cost   13)
<XY  6.41, 4.87 @ -15.0 deg> (fwd  1, str -1, cost   14)
<XY  6.79, 4.75 @ -20.0 deg> (fwd  1, str -1, cost   15)
<XY  7.16, 4.60 @ -25.0 deg> (fwd  1, str -1, cost   16)
<XY  7.51, 4.41 @ -30.0 deg> (fwd  1, str -1, cost   17)
<XY  7.17, 4.63 @ -35.0 deg> (fwd -1, str  1, cost   18)
<XY  6.86, 4.87 @ -40.0 deg> (fwd -1, str  1, cost   19)
<XY  6.56, 5.14 @ -45.0 deg> (fwd -1, str  1, cost   20)
<XY  6.30, 5.43 @ -50.0 deg> (fwd -1, str  1, cost   21)
<XY  6.05, 5.75 @ -55.0 deg> (fwd -1, str  1, cost   22)
<XY  5.81, 6.07 @ -50.0 deg> (fwd -1, str -1, cost   23)
<XY  5.54, 6.36 @ -45.0 deg> (fwd -1, str -1, cost   24)
<XY  5.25, 6.63 @ -40.0 deg> (fwd -1, str -1, cost   25)
<XY  4.93, 6.87 @ -35.0 deg> (fwd -1, str -1, cost   26)
<XY  4.59, 7.09 @ -30.0 deg> (fwd -1, str -1, cost   27)
<XY  4.24, 7.27 @ -25.0 deg> (fwd -1, str -1, cost   28)
<XY  4.61, 7.12 @ -20.0 deg> (fwd  1, str  1, cost   30)
<XY  4.99, 7.00 @ -15.0 deg> (fwd  1, str  1, cost   31)
<XY  4.60, 7.08 @ -10.0 deg> (fwd -1, str -1, cost   32)
<XY  4.99, 7.03 @  -5.0 deg> (fwd  1, str  1, cost   34)
<XY  4.60, 7.05 @  -0.0 deg> (fwd -1, str -1, cost   35)
```

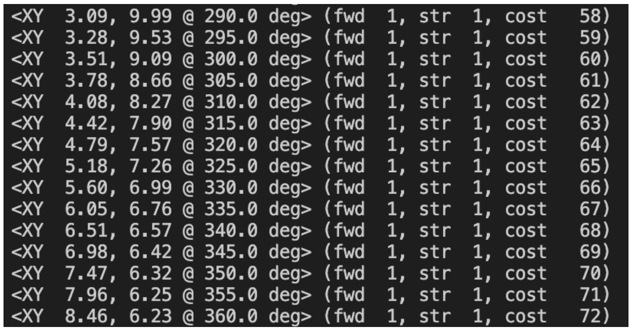
Problem 5 (Changing Cost - Moving Over) - 16 points:

part a The following cost was used to create a path with straighter segments:

$$cost = c_{steer}|steer| + c_{reverse}N_{reversals}$$

Where $c_{steer} = 1$ and steer can either be -1,0, or 1 as specified in 1b of the problem set. In other words, the cost increases when the robot is not going straight. The $c_{reverse}N_{reversals}$ part was keep so that the robot still moves in a similar fashion. $c_{reverse}$ was set to 100. With this cost the following path and cost values were produced (note that only the costs of the last few nodes are shown, but the whole path is shown):



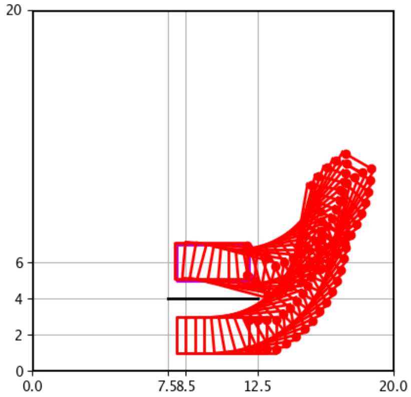


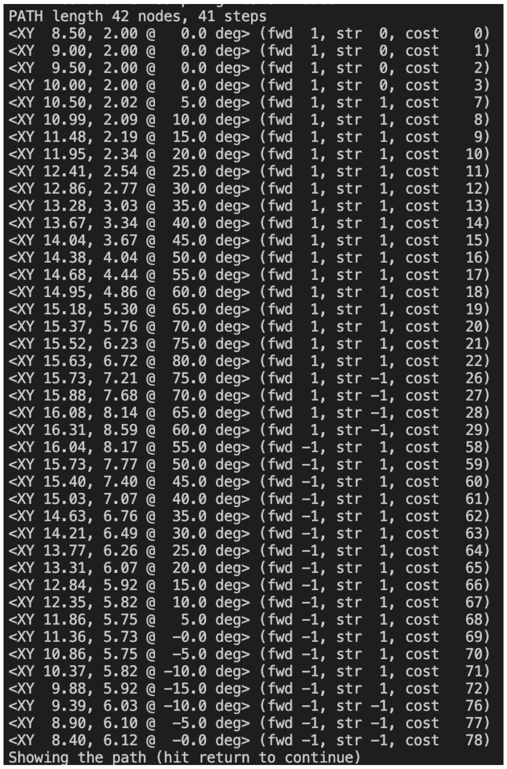
```
<XY  3.09, 9.99 @ 290.0 deg> (fwd  1, str  1, cost   58)
<XY  3.28, 9.53 @ 295.0 deg> (fwd  1, str  1, cost   59)
<XY  3.51, 9.09 @ 300.0 deg> (fwd  1, str  1, cost   60)
<XY  3.78, 8.66 @ 305.0 deg> (fwd  1, str  1, cost   61)
<XY  4.08, 8.27 @ 310.0 deg> (fwd  1, str  1, cost   62)
<XY  4.42, 7.90 @ 315.0 deg> (fwd  1, str  1, cost   63)
<XY  4.79, 7.57 @ 320.0 deg> (fwd  1, str  1, cost   64)
<XY  5.18, 7.26 @ 325.0 deg> (fwd  1, str  1, cost   65)
<XY  5.60, 6.99 @ 330.0 deg> (fwd  1, str  1, cost   66)
<XY  6.05, 6.76 @ 335.0 deg> (fwd  1, str  1, cost   67)
<XY  6.51, 6.57 @ 340.0 deg> (fwd  1, str  1, cost   68)
<XY  6.98, 6.42 @ 345.0 deg> (fwd  1, str  1, cost   69)
<XY  7.47, 6.32 @ 350.0 deg> (fwd  1, str  1, cost   70)
<XY  7.96, 6.25 @ 355.0 deg> (fwd  1, str  1, cost   71)
<XY  8.46, 6.23 @ 360.0 deg> (fwd  1, str  1, cost   72)
```

part b The following cost was used to create allow the car to pull forward and then back up into the neighbor spot:

$$cost = c_{step}N_{step} + c_{steer}N_{steeringAnglesChanges} + c_{reverse}N_{reversals}$$

Where $c_{step} = 1$, $c_{steer} = 3$, $c_{reverse} = 25$.With this cost the following path and cost values were produced:





```
PATH length 42 nodes, 41 steps
<XY  8.50, 2.00 @   0.0 deg> (fwd  1, str  0, cost    0)
<XY  9.00, 2.00 @   0.0 deg> (fwd  1, str  0, cost    1)
<XY  9.50, 2.00 @   0.0 deg> (fwd  1, str  0, cost    2)
<XY 10.00, 2.00 @   0.0 deg> (fwd  1, str  0, cost    3)
<XY 10.50, 2.02 @   5.0 deg> (fwd  1, str  1, cost    7)
<XY 10.99, 2.09 @  10.0 deg> (fwd  1, str  1, cost    8)
<XY 11.48, 2.19 @  15.0 deg> (fwd  1, str  1, cost    9)
<XY 11.95, 2.34 @  20.0 deg> (fwd  1, str  1, cost   10)
<XY 12.41, 2.54 @  25.0 deg> (fwd  1, str  1, cost   11)
<XY 12.86, 2.77 @  30.0 deg> (fwd  1, str  1, cost   12)
<XY 13.28, 3.03 @  35.0 deg> (fwd  1, str  1, cost   13)
<XY 13.67, 3.34 @  40.0 deg> (fwd  1, str  1, cost   14)
<XY 14.04, 3.67 @  45.0 deg> (fwd  1, str  1, cost   15)
<XY 14.38, 4.04 @  50.0 deg> (fwd  1, str  1, cost   16)
<XY 14.68, 4.44 @  55.0 deg> (fwd  1, str  1, cost   17)
<XY 14.95, 4.86 @  60.0 deg> (fwd  1, str  1, cost   18)
<XY 15.18, 5.30 @  65.0 deg> (fwd  1, str  1, cost   19)
<XY 15.37, 5.76 @  70.0 deg> (fwd  1, str  1, cost   20)
<XY 15.52, 6.23 @  75.0 deg> (fwd  1, str  1, cost   21)
<XY 15.63, 6.72 @  80.0 deg> (fwd  1, str  1, cost   22)
<XY 15.73, 7.21 @  75.0 deg> (fwd  1, str -1, cost   26)
<XY 15.88, 7.68 @  70.0 deg> (fwd  1, str -1, cost   27)
<XY 16.08, 8.14 @  65.0 deg> (fwd  1, str -1, cost   28)
<XY 16.31, 8.59 @  60.0 deg> (fwd  1, str -1, cost   29)
<XY 16.04, 8.17 @  55.0 deg> (fwd -1, str  1, cost   58)
<XY 15.73, 7.77 @  50.0 deg> (fwd -1, str  1, cost   59)
<XY 15.40, 7.40 @  45.0 deg> (fwd -1, str  1, cost   60)
<XY 15.03, 7.07 @  40.0 deg> (fwd -1, str  1, cost   61)
<XY 14.63, 6.76 @  35.0 deg> (fwd -1, str  1, cost   62)
<XY 14.21, 6.49 @  30.0 deg> (fwd -1, str  1, cost   63)
<XY 13.77, 6.26 @  25.0 deg> (fwd -1, str  1, cost   64)
<XY 13.31, 6.07 @  20.0 deg> (fwd -1, str  1, cost   65)
<XY 12.84, 5.92 @  15.0 deg> (fwd -1, str  1, cost   66)
<XY 12.35, 5.82 @  10.0 deg> (fwd -1, str  1, cost   67)
<XY 11.86, 5.75 @   5.0 deg> (fwd -1, str  1, cost   68)
<XY 11.36, 5.73 @  -0.0 deg> (fwd -1, str  1, cost   69)
<XY 10.86, 5.75 @  -5.0 deg> (fwd -1, str  1, cost   70)
<XY 10.37, 5.82 @ -10.0 deg> (fwd -1, str  1, cost   71)
<XY  9.88, 5.92 @ -15.0 deg> (fwd -1, str  1, cost   72)
<XY  9.39, 6.03 @ -10.0 deg> (fwd -1, str -1, cost   76)
<XY  8.90, 6.10 @  -5.0 deg> (fwd -1, str -1, cost   77)
<XY  8.40, 6.12 @  -0.0 deg> (fwd -1, str -1, cost   78)
Showing the path (hit return to continue)
```

Problem 10 (Time Spent) - 4 points:

I spent about 3 hours on the problem set.