Algorithmic Robotics Project 2

1. In this project, wrote collision checking algorithms (Implemented in isValidPoint, isValidCircle, isValidSqaure)to check if the three different shaped robots (Point, Circle, Square) are valid in a 2-D plane work environment.

2. Wrote the functions to check the collision between the robot and obstacles. Three different types of robots used for this purpose represented a point, circle and square. The point robot is a 1-D robot and is represented by co-ordinates. For the point robot the C-space and workspaces were the same. Since the geometry was simple, it is just ti compare whether the co-ordinates of the robot lied between the co-ordinates of the obstacle. For circle and square robots were 2-D robots and in these cases the C-space and workspace were different in comparison to each other. The circle robot was represented by its center and radius. The workspace of the circle robot can be thought of as extended by the radius with rounded edges. Lastly, we worked with a square robot, which was represented by the center co-ordinates, the side length and a rotation angle θ. The configuration code for this robot was complex as compared to the other two. We perform collision checking in the workspace. The obstacles are rectangles represented by the the height, width and the left bottom vertex co-ordinates.

3.
Point Robot: The collision checking for this robot was the easiest, just had to check whether the co-ordinates of this robot lied in a range. The range was obtained from the co-ordinates and dimensions of the obstacles.

Circle Robot: The circle robot was a bit difficult as compared to point robot, but the approach was quite similar. The center and radius was know for circle robot. Had to verify whether the center lied in the range of co-ordinates of the obstacle. Calculated and checked if the Euclidean Norm between the center and vertices of the obstacle and compared it with the radius of the circle.

Square Robot: This was difficult problem to solve.
case 1: when the any of the vertex of the rotated square lies inside the obstacle, for this used the isValidPoint function to determine this, also checked if the centroid of the sqaure lies in the obstacle.

case 2: when the any of the vertex of the anticlockwise rotated obstacle w.r.t to square robot centroid lies inside the non rotated translated square robot, for this also used the isValidPoint function to determine.

No line intersection needed to performed. Got 100% accuracy in all the conditions.

4.
Point Robot: 3/10 30 min.
Circle Robot: 5/10 1 hr.
Square Robot: 8/10 6 hr.
Report: 2/10 1 hr.