**CSE4340/CSE5349, Spring 2015, Final Project**

Due ***on Blackboard*** **BEFORE CLASS** by May 7, 2015, **1:59** pm central time

**Final Project Description:**

For the final project, you will build a new StarL app. This will be a team project, with a maximum of 2 people per team (1 team of 1 or 3 will be allowed in the event of an odd number).

Teams will not be assigned, so you should pick your own team. The app you develop should solve the following problem (see end of document for pictorial description):

There are K sets of waypoints that need to be visited in order (sequence), by N robots.

**Your app must also meet the following minimum requirement:**

1. Each robot must visit at least 1 waypoint in each set. Assume each set of waypoints 𝑊𝐾 has at least N waypoints.
2. Each set of waypoints must be visited in the proper order (it is ok if a bot travels over a waypoint not in the set currently being visited, but it should not be counted as visited).
3. All waypoints must be visited.

You will lose points if your app does not meet the above requirement in addition to being disqualified from the competition. All apps meeting these requirements will receive full credit.

You have been given a minimal template for the app, which contains a test environment with obstacles and sets of waypoints. Your app will be tested in a similar but different environment.

**Competition:**

A competition will be held during the class final exam time. Each project will be run and timed. The winner will be the team with lowest total time (including penalties). The following describes in detail how the completion will be scored.

1. *Time Objective*: visit all waypoints in the minimum amount of time,
   * Note: this contains a **progress** requirement! If your app deadlocks, time goes to infinity, so you lose. For this, may require a **safety** property.
2. *Robot Collision*: minimize the total number of collisions between robots. Receive a time penalty for each robot-robot collision. This is a **safety** requirement!
3. *Obstacle Collision*: minimize the total number of collisions between robots with obstacles in the environment. Receive a time penalty for each robot-obstacle collision. This is a **safety** requirement!

Rules for the competition will be as follows.

* Test environment will be provided
  + Includes waypoints and obstacles
  + Competition environment will be different, but similar, and not revealed until competition
* Cheating: if we detect cheating (modifying things that shouldn’t be, etc.), will disqualify
  + Likewise if we detect plagiarism, etc.
* No ties:
  + Winner has the minimum overall time while meeting all rules and requirements
  + If there is a time tie, we will take the one with the fewest collisions as the winner
  + We will flip a coin to pick winner if there are still ties

Your class participation points (10% of total grade, see syllabus) will be awarded for competing, so everyone should plan on participating in the competition.

**Awards:**

* 1st place: guaranteed A in course
* 2nd place: +100 points on final project grade
* 3rd place: +50 points on final project
* Everyone that competes: 10% class participation points

**Additional Opportunity for Bonus Points:**

Anyone who successfully deploys to hardware (using Android tablets/phones and AR Drones) will receive 30 extra points on the final project. This may require developing an interface between Android tablet and AR Drone via WiFi. Anyone interested should speak with Dr. Johnson for more details.

**Things to Think About:**

* How to prevent deadlocks?
  + Robots may have to move if they’re in the way (recall yellow example)
  + This may also happen if they’re just staying still
* How can we decompose this problem?
* What fundamental mobile/distributed systems problems does it involve?
  + Interesting variations of mutual exclusion: may need mutual exclusion, but also want ***speed***
  + Have to carefully balance speed and getting into deadlocks
  + Have to carefully balance speed and collisions
* Speed strategies
  + Maybe in some instances it is ideal to have a central coordinated (leader)
  + Maybe it’s better to be greedy (each robot goes to nearest waypoint in current set to visit)
  + Maybe it’s sometimes best to simply have some collisions
* What other subproblems are effectively encoded?
  + Traveling salesman, …
* Solving strategy
  + First pass: may make sense to not worry about speed and/or collisions, just get working
  + Second pass: start to refactor and think about how to incorporate speed, collisions, deadlocks, etc.

**References:**

The following references may also be useful, but are not totally necessary to complete the assignment.

<https://github.com/verivital/starl-uta/blob/master/StarLTrafficSignApp.pdf>

A somewhat outdated overview is available here, which is useful for understanding how the code is organized and subdivided:

<https://github.com/verivital/starl-uta/blob/master/Using%20the%20StarL%20Framework.pdf>

A recent paper on StarL is available here:

StarL: Towards a Unified Framework for Programming, Simulating and Verifying Distributed Robotic Systems, Yixiao Lin, Sayan Mitra, February 2015.

<http://arxiv.org/abs/1502.06286>

A detailed and slightly outdated overview of StarL is available in this thesis:

StarL for programming reliable robotic networks, Adam Zimmerman, Master’s Thesis, December 2012.

<https://www.ideals.illinois.edu/handle/2142/42380>

**Pictorial Example:**











































