# HW7 Part A

1. Consider the use case (application) of a Robot driving a car. In this context, what is RL? How can the ADP and TD methods be used for this? What about the Active RL method?

   **Solution:**

   - Reinforcement learning is a learning mechanism that learns how to map states to actions
     in order to maximize rewards. The learner is not told which actions to take but must try
     to find which ones will produce the greatest return.
   - In the context of Robot driving a car (autonomous vehicles), the agent needs to map actions : acceleration, deceleration, and steering to states from monitoring the environment and realize which new state could receive a maximum reward.
   - **TD methods** can be used in autonomous vehicles as they use experience to predict actions without the need of modeling the car's environment. Also the system needs to wait only one time step, not delaying all learning until the end of the trip (driving).
   - Unlike TD, **ADP** is a model based approach and requires the transition model of the environment. It estimates the utility of a state as a sum of reward for being in that state and the expected discounted reward of being in the next state.
   - By intelligently choosing which rewards to observe in the autonomous car environment, **active RL** can learn more efficiently about the reward function and maximize the discounted sum of rewards.

2. What are Bandits methods, how and why are they used? Think of an example application

   **Solution:**

   Bandit methods are in which an agent has to select actions in order to maximize its cumulative reward in the long term. In each round, the agent receives some

information about the current state, then it chooses an action based on this information and the experience gathered in previous rounds. At the end of each round, the agent receives the reward associated with the chosen action. At each time step, the agent takes an action on the environment based on its policy $\pi(a_t|s_t)$, where $s_t$ is the current observation from the environment, and receives a reward $r_{t+1}$ and the next observation $s_{t+1}$ from the environment. The goal is to improve the policy so as to maximize the sum of rewards (return).

**Application of Bandit Methods: A/B Testing**

In A/B testing, bandit methods utilize machine learning to learn from data collected during the test and dynamically boost visitor allocation in favor of better-performing variations. This means that ineffective versions receive less and less traffic allocation over time. MAB maximizes the total number of conversions during the test, unlike standard A/B tests. Because the focus is on conversions and determining the actual conversion rates, statistical certainty takes a back seat (of all variations, including the worst performing ones). Most traditional A/B tests are always in 'exploration' mode by design — after all, identifying statistically meaningful results is their raison d'être, hence the constant exploration. The goal of an A/B test is to determine the exact conversion rate of two different versions. MAB gives A/B Testing a new dimension: exploitation. Because MAB's goal is to "maximize conversions and profit," exploitation and exploration run in tandem, like a train track. Imagine the algorithm exploring at a rate of many visitors per second, arriving at constantly shifting winning baselines, and dynamically allocating the majority of your traffic to the variant that has a better chance of winning at that moment.

3. Answer from our textbook Norvig & Russell page 858 Question 21.1 - this is a Python
implementation.

   *21.1 Implement a passive learning agent in a simple environment, such as the 4 × 3 world.*
   *For the case of an initially unknown environment model, compare the learning performance*
   *of the direct utility estimation, TD, and ADP algorithms. Do the comparison for the optimal*
   *policy and for several random policies. For which do the utility estimates converge*

*faster?*
*What happens when the size of the environment is increased? (Try environments with and*
*without obstacles.)*

**Solution: q3_passive_agent**

4. Please implement a Q Learning algorithm similar to this tutorial: https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/ but to use the maze problem we learned in class (see Q-Learning Example.docx) and prove your implementation using this data set.

   **Solution: Check Directory - /q4_maze**