- Local search operators to try out
    - Need one general operator but it should generate reasonably sized moveset
    - Is it required that every tour in the moveset is a better solution than the current tour?
    - Why do we need a set of states as moveset? A single state to move next would suffice right?
        - Use the operation specified here http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/ and generate a moveset of fixed size
        - Just realized that its better to use an operator that with greater probability lead to better choices, but can also generate bad ones
            - Like choosing the longest 2 edges and crossing them up
        - Generating an initial move to start with ?
            - Start with city A and move to the nearest unvisited city of the current tour. This is greedy approach
- When to stop simulated annealing ?
    - When after a certain number of steps, and when you stop seeing improvement in the solution
    - No need of random restart for this assignment
- Cooling strategies
    - Exponential schedule
        - $T (t) = T0\alpha t$ where $0 < \alpha < 1$
    - Linear schedule
        - $T (t) = T0 - \eta t$
    - Logarithmic schedule
- How do we implement 'selecting with probability P'?
- Resources
    - For TSP route online https://neos-server.org/neos/solvers/co:concorde/TSP.html
    - http://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/
    - https://github.com/chncyhn/simulated-annealing-tsp
    - Java Implementation http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6
    - For cooling strategy http://iopscience.iop.org/article/10.1088/0305-4470/31/41/011/pdf
    - schedule 2 from here http://what-when-how.com/artificial-intelligence/a-comparison-of-cooling-schedules-for-simulated-annealing-artificial-intelligence/
    - https://inst.eecs.berkeley.edu/~cs188/fa08/section_handouts/section3_handout_solutions.pdf
        - Has answer for question 2 - both parts a and b
    - Question 3

- https://www.cs.rochester.edu/~brown/242/answers/07Mid.pdf has answers
- Also read this for more hints http://inst.cs.berkeley.edu/~cs188/sp11/section_handouts/section3_solution.pdf
- Part a - treat it as multiplayer case and each node's utility is a vector of size 2

- Q4
  - What the basic player does?
    - a rudimentary basic player utilizing minimax search
    - 4-ply depth search
  - What we should do?
    - You will implement a better agent that uses
    - alpha-beta pruning and better evaluation functions
    - The only file you need to modify is `implementation.py`
  - Testing our player
    - See the assignment on how to run tests
  - Must contain student ID and a name for our agent
  - To compete in the tournament
    - See assignment on what to do
  - Report.pdf explain your 'better evaluation function'
  - Upload only implementation.py and report.pdf
- Understand the game
  - Gravity – a token falls to the lowest unoccupied cell in the column
  - Wins when 4 tokens lined up vertically, horizontally and diagonally
- My solution
  - Something beyond 4-ply depth ??
- Resources for evaluation function
  - http://web.mit.edu/sp.268/www/2010/connectFourSlides.pdf
  - https://www.quora.com/What-is-the-winning-strategy-for-the-first-player-in-Connect-Four-games
  - https://inst.eecs.berkeley.edu/~cs188/fa09/section_handouts/connect4_solutions.pdf
  - https://www.gamedev.net/forums/topic/225611-connect-4-evaluation/
  - Victor's paper http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf
  - https://cs.stackexchange.com/questions/13453/trying-to-improve-minimax-heuristic-function-for-connect-four-game-in-js
  - https://www.reddit.com/r/dailyprogrammer/comments/3fva66/20150805_challenge_226_intermediate_connect_four/
- After a day of searching for evaluation function I have realized that its better to implement AB pruning first