Project 1: Search

Question 1:

- The order is as expected, deepest nodes are expanded first and within same deepness it is randomized (based on which successor is returned first).
- No, he does not, he goes to the goal on the shortest path that was found through expansion.
- DFS does not find the least cost solution. Following the first quasi-random path (again, based on the first successor returned) it finds a solution, but it does not have to be optimal -> complete, not optimal

Question 2:

- BFS is complete and optimal for non-varying step cost. A simple explanation is that all paths are expanded in parallel until the first path reaches the goal -> this must be the shortest path.

Question 3:

- If step-cost varies BFS is not optimal. Just searching in parallel could lead to a short path with high cost. Hence UCS is needed. Since a Priority-Queue is used for UCS, nodes with low cost are prioritized in expansion. This leads to completeness, optimality and is the difference to BFS.

Question 4:

- DFS does not find the optimal solution. BFS and UCS do but with more node expansions than A*.

Question 5:

- DFS does not find the optimal solution. BFS does but to do so it works on all paths in parallel randomly. A* reduces node expansion by searching in the right direction via the heuristic.

Question 6:

- Given Pacman's position and the coordinates of the corners one can implement a Manhattan heuristic. This can never be larger than the actual cost to the goal -> admissible. Additionally, it is consistent because it cannot be reduced more than a step-cost per step. It is crucial to return the Manhattan distance to the furthest corner (visited last), since after visiting that the goal state is reached.

Question 7:

- Given the complexity of this problem the Manhattan heuristic is not effective enough. We can make use of the implemented BFS to find the distances to all food elements, of these we can then return the max distance as a consistent heuristic.

Question 8:

- This greedy search is not optimal. We can think of a 11x1 maze where Pacman starts in the center. To the right we have 5 food elements, to the left 1, not directly next to Pacman but with a gap of one empty field. Pacman will now run to the right, eating 5 food elements, then having to travel all the way back to the last food. There is an infinite number of problems that make this agent inefficient.