Q1 – DFS
Data Structure used: Stack
The information being pushed on the stack: Coordinates of the successor's nodes of the last node popped and the corresponding direction needed to get to these children node.

Q2 – BFS
Data Structure used: Queue
The information being added to the Queue: Coordinates of the successor's nodes of the last node popped and the corresponding direction needed to get to these children node.

Q3 – Uniform Cost Search
Data Structure Used: Priority Queue Weighted on the basis of the cost from start state to the current node being pushed to the queue.
Information being added to the Priority Queue items: tuple of the coordinates of nodes, list of paths followed to get to these nodes from the start state and the cost to get to these nodes from start state for each successor node of the last popped node

Q4 - A*
Data Structure Used: Priority Queue Weighted on the basis of the sum of cost from start state to the current node being pushed to the queue and the heuristic cost for this node.
Information being added to the Priority Queue items: tuple of the coordinates of nodes, list of paths followed to get to these nodes from the start state and the cost to get to these nodes from start state, for each successor node of the last popped node

Q5 - Corners Problem
State - At any point during the corners problem, the state of the PAC Man includes the current coordinates of the PAC man and a list of corners that have already been covered up until that state is reached. Each time a corner is reached in a state, its coordinates are appended to this list.
Maintaining a list of corners serves two purposes.
1. It helps in revisiting the coordinates that have already been covered (because of state change on adding a new item to the list), which would not have been possible in case only coordinates were used to represent the state.
2. It helps us to keep track of how close we are to the goal state.

Goal State: If the size of the list of corners becomes equal to the number of corners in the problem(4 in our case), goal state is reached.

Q6 - Heuristic for Corners Problem
First, we find all the corners that have not been visited yet and then find cost to reach the nearest corner from the current state. Then we update the current state to this corner state and calculate the cost to its next nearest neighbor. This process of updating state and calculating costs is repeated until all remaining corners are explored.
Heuristic distance = (Cost to reach the nearest corner from current state) + (cost to reach other corners as explained above)
This heuristic is admissible as we are using Manhattan distance to calculate the cost at any given state. The first term in our heuristic evaluation equation ensures that the heuristic is consistent.

Q7- Heuristic Food Search Problem
Used the Maze distance function to find out the unconsumed food that is farthest from the current state. This would be the minimum distance that will need to be traveled by the Pacman. As the Pacman will have to travel more distance to consume other food pixels, our heuristic is admissible and consistent