# Week 5 - Hands On Exercise - Review Questions

ASD103A-21: Object-Oriented Data Structures using Python, Part1

Prof: Debsankar Mukhopadhyay

Delvis Soto-Soto

Software Developer

Question 1: Select any three options (A, B, C) below that represent lists, and in your own words, explain why or why not they can (or cannot) be considered as lists.

Customers waiting in a checkout line (A): Yes, this can be considered a list. It's an ordered collection of people waiting and served in order.

A deck of playing cards (B): No, this is not a list. It's more like an array or stack with a specific order.

A file directory system (C): No, this is not a list. It's a hierarchical structure, not a linear list.

Summary:

A: Yes (Checkout line is a list).

B: No (Deck of cards is not a list).

C: No (File directory system is not a list).

Question 2: Describe each graph's terminology (edge, vertex and so on).

Vertex (or Node): A fundamental unit in a graph. It represents a point or entity in the graph.

Edge: A connection or link between two vertices. It represents a relationship between the entities represented by the connected vertices.

Directed Graph (DiGraph): A graph where edges have a direction, indicating a one-way relationship between vertices.

Undirected Graph: A graph where edges have no direction, indicating a two-way relationship between vertices.

Weighted Graph: A graph where each edge has an associated weight or cost.

Cycle: A sequence of vertices in which the first and last vertices are the same, forming a closed loop.

Connected Graph: A graph in which there is a path between every pair of vertices.

Question 3: Describe the logical flow behind Dijkstra’s Algorithm.

Dijkstra's Algorithm is a method for finding the shortest path between two nodes in a graph. The logical flow includes the following steps:

Initialization: Start with the initial node and set its distance to zero. Set the distances of all other nodes to infinity. Create a priority queue or min-heap to manage the nodes based on their tentative distances.

Relaxation: For the current node, consider all of its neighbors and calculate their tentative distances through the current node. If a shorter path is found, update the neighbor's distance.

Selection: Select the node with the smallest tentative distance from the priority queue. This node becomes the "current node."

Repeat: Repeat steps 2 and 3 until the destination node is marked as "visited" or the priority queue is empty.

Optimal Path Reconstruction: Once the destination node is reached, backtrack from the destination node to the source node, selecting the nodes with the smallest tentative distances along the way. This gives the shortest path.