1. Perform Dijkstra’s shortest path algorithm on this graph to find the shortest distance from C to E. Include the following:

(a) The final distance and predecessor values for all finalized vertices.
   **Answer:** A: 4, C; B: 6, A; C: 0; D: 11, A; E: 11, B; F: 6, C

(b) The order in which the vertices are finalized.
   **Answer:** C, A, B, D, E, F

(c) The list of vertices in the shortest path from C to E and the length of that path.
   **Answer:** [C, A, B, E] 11

2. When is a vertex’s sum weight finalized in Dijkstra’s algorithm?
   **Answer:** When it non-finalized and it has the current lowest cost.

3. What role does the priority queue play in finding the shortest path? When do we use it?
   **Answer:** It keeps track of the non-finalized vertices. It has functions to enqueue a new vertex, and a dequeue function that removes the least cost vertex.

4. What data structure/s can be used to represent a weighted graph?
   **Answer:** Adjacency list that maps vertices to neighbors with weights, i.e.:
   A: (B,2), (C,4), (D,7), (F,5) B: ...
   The priority queue is used to select the minimum unmarked vertices. In class we implement it using a standard list (enqueue to the front, dequeue linearly scans and removes the smallest element). We aren’t going to talk about the optimal implementation which is a heap (wk 9).
   Note: If you want to design classes for this after discussion that is good.

5. What is the complexity for the “naive” implementation of the priority queue we used in lecture?
   **Answer:** Enqueue: O(1) - Insert element at head of list. Dequeue: O(N) - Search and remove/return smallest element from list.
6. What is the overall time complexity of Dijkstra’s algorithm?

**Answer:** Assume E edges and V vertices, it is $O(|E||V|)$ (see lecture writeup for more details ;)