Complete hw by giving answers to the questions below, then upload the document in Canvas:

Exercise 4.3 Design an algorithm that reverses a doubly-linked list, without creating any new node objects.

Answer:

Exercise 4.6. Refactor (rewrite) our queue data structure so that, instead of using a doubly-linked list, it instead uses two stacks. The time complexity of all operations in your list should match those in Table 4.7 (although some of the time bounds for your structure may be amortized). Which variant do you prefer, and why?

Answer:

Exercise 4-7. Graph modeling. For each of the following kinds of digital information, explain how that information could be stored in a graph data structure. Explain specifically what each vertex  
corresponds to, what each edge corresponds to, what each vertex label (if any) represents,  
what each edge label (if any) represents, and whether the graph is directed, undirected, or  
mixed. Draw a sketch of an example graph with a handful of vertices and edges.  
(a) A street map including streets and intersections.  
(b) A social network, such as the web of people and friendships present on a social network  
site such as Facebook or LinkedIn.  
(c) The set of courses and prerequisite relationships for your major.

Answer:

4.11 Do some Internet research to find out which kind of self-balancing binary search tree (e.g.  
AA tree, AVL tree, etc.) is used in the C++ standard template library (STL) map and set .  
As far as you can tell, why is that kind of tree used over the alternatives listed in Section 4.7.

Answer:

Exercise 5.5 Sort the characters of the string “SEQUENCE” using pure selection sort; show your work.

Answer:

Exercise 5.6 Sort the characters of the string “SEQUENCE” using in-place selection sort; show your work.

Answer:

Exercise 5.9 Suppose that selection sort is applied to a list that is already in sorted order. What is the time complexity of the algorithm this case?

Answer:

Exercise 5.10 Design a sorting algorithm whose time complexity is as follows:  
(a) when the input happens to be already non-decreasing, the algorithm takes only O(n) time;  
(b) likewise, when the input happens to be already non-increasing (i.e. reverse-sorted), the algorithm takes only O(n) time;  
(c) but in any other situation, the algorithm may take O(n2) time.  
Justify that your algorithm meets these efficiency goals.

Answer: