Question 1 of 3

Recall that, so far, we've seen at least two ways to store a sequence of values: arrays and linked lists.

a) What's one advantage that using an array has over using a linked list?
b) What's one advantage that using a linked list has over using an array?

Answers

a) Answers may vary. Possibilities include:
    i) Random access. In an array, you can access any element in the array in constant time. In a linked list, it takes linear time to access a particular element.
    ii) Uses less memory. In an array, you only need to store the values you care about representing in your sequence. In a linked list, you additional need to store pointers to nodes.
    iii) Binary search. Because of random access, arrays lend themselves to using binary search if the array is sorted. But even if a linked list is sorted, you can't perform binary search efficiently because there's no ability to access the middle element in constant time.
b) Answers may vary. Possibilities include:
    i) Size can change. In an array, the size is fixed, so it can't easily grow or shrink — if you need to store more data, you might need to use a different location in memory and copy all the data over. In a linked list, it's easy to insert and remove nodes from the list to change its size.
    ii) Insertion. With a linked list, it's easier to insert a node in between two other nodes, simply by reassigning the next pointers to be in the correct order. In an array, you can't simply insert a new value in between two existing values.

Question 2 of 3

Recall from lecture that looking up a value in a hash table takes O(n) (linear) time. We also saw that looking up a value in a linked list takes linear time, too. If both data structures require linear time to look up a value, why might we prefer a hash table nevertheless?

Answers

Even though both take O(n) time in the worst case, it's likely that actual lookups will take less time with a hash table on average. Since the data in a hash table is split across multiple different linked lists, each individual linked list will be shorter, and thus take fewer steps to search through.

Question 3 of 3

Recall that in lecture, we saw how Jack could use a queue or a stack to organize his clothing. Consider other real-world scenarios where you might want to store physical objects or data using a queue or a stack.

a) Give an example of a real-world scenario where you might want to store physical objects or data in a stack. Why is a stack the preferred data structure to use for that use case?
b) Give an example of a real-world scenario why you might want to store physical objects or data in a queue. Why is a queue the preferred data structure to use for that use case?

Answers

a) Answers may vary. Possibilities include:
    i) Papers in a drawer: any time you want to store paper in the drawer, add it to the top of the stack. By using a stack, you'll always have access to the most recent papers you've been working with most easily, which are likely the ones you'll care about accessing.
    ii) Internet search history: pages visited are stored with the most recent items at the top, which is preferred because the most recent pages are most likely the ones the user would want to visit again.
    iii) Missed calls or voicemails in a phone application: The most recently missed calls and most recent voicemails show up at the top of the list, which is preferred because it means the missed calls that are most relevant to the user are likely the most recent.
b) Answers may vary. Possibilities include:
    i) RSVP system: when letting people register for a limited number of tickets for an event, you'd likely want to organize registrants in a queue. Doing so ensures that the first person to register receives the first ticket; if the data were organized into a stack, then the last people to register might receive the tickets, rather than the first registrants.
    ii) Downloads list: If you're downloading many items at once, you'd likely want to order those downloads such that the earlier items download first. The queue is preferred because it ensures that older downloads aren't stalled due to newer downloads.