CS152: Data Structure

Week 5 Knowledge Check

Write the full name of all collaborators inside this box

Discuss these questions and write your answers in the space provided below it.

1. Assume that the position of an item to be removed from a singly linked structure has been located. State the run-time complexity for completing the removal operation from that point.

2. Can a binary search be performed on items that are in sorted order within a singly linked structure? If not, why not?

3. Suggest a good reason that Python list uses an array rather than a linked structure to hold its items

4. What advantage does a circular linked structure with a dummy header node give the programmer?

5. Describe one benefit and one cost of a doubly linked structure, as compared to a singly linked structure.

6. Suppose you start with an empty circular linked list that uses a dummy header node (as in the code below). You then call the `insert` method three times with the values 10, 20, and 30 (in that order):

```
# Recall the same circular linked structure implementation shown in the slides

class Node:
    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next

class CircularLinkedList:
    def __init__(self):
        # Create a dummy header node that points to itself
        self.header = Node()
        self.header.next = self.header

    def insert(self, data):
        # Insert new node at the end (before header)
        new_node = Node(data, self.header)
        probe = self.header
        while probe.next != self.header:
            probe = probe.next
        probe.next = new_node

# Using the circular linked structure
clist = CircularLinkedList()
clist.insert(10)
clist.insert(20)
clist.insert(30)
```

- Draw the resulting circular linked structure, clearly labeling:
  - The dummy header node,
  - All data nodes (with their values),
  - All next pointers.
  - Show the direction of the links and indicate which node is the header.