Quiz 18

Due Jun 7 at 11:59pm **Points** 14 **Questions** 14
Available until Jun 7 at 11:59pm **Time Limit** None

Instructions

Answer the following questions.

This quiz is no longer available as the course has been concluded.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	9 minutes	14 out of 14

Score for this quiz: **14** out of 14
Submitted Jun 7 at 6:20pm
This attempt took 9 minutes.

Question 1 1 / 1 pts

Doubly-linked list is a list where

Correct! ●

every node has a link to the next node, and another link to the previous node

- the list has twp nodes
- every node has a single link to the next node

Question 2

1 / 1 pts

In appending a node to an empty doubly-linked list which pointer gets updated?

- Tail
- Head
- [✓ Both head and tail]

Correct!

Question 3

1 / 1 pts

In appending a node to a non-empty doubly-linked list which pointer gets updated?

- new node's next pointer
- Head
- [✓ Tail] Correct!

Question 4

1 / 1 pts

In prepending a node to an empty doubly-linked list which pointer gets updated?

- Tail
- [✓ Both head and tail] Correct!
- Head

Question 5

1 / 1 pts

In prepending a node to a non-empty doubly-linked list which pointer gets updated?

Correct!

[✓ Head]

new node's prev

Tail

Question 6

1 / 1 pts

Consider the following doubly-linked **list** in order: 3, 6, 8.

Which pointers get updated in **ListInsertAfter(list, node 8, node 5)** is invoked.

○ node 5 -> prev

○ tail -> next

Correct!

[✓ node 5 -> prev and tail -> next]

Question 7

1 / 1 pts

Consider the following doubly-linked **list** in order: 3, 6, 8.

Which pointers do not get updated in **ListInsertAfter(list, node 6, node 5)** is invoked?

○ node 6 -> next

○ node 5 -> prev

○ node 5 -> next

Correct!

[✓ node 6 -> prev]

Question 8

1 / 1 pts

Consider the following dubly-linked list of numbers in order: 3, 6, 8, 2.

Which pointers get updated if **ListRemove(list, node 6)** is invoked?

Correct!

- [✓ node 3 -> next and node 8 -> prev]
- node 6 -> next and node 6 -> prev
- node 3 -> next and node 6 -> prev
- node 6 -> next and node 8 -> prev

Question 9

1 / 1 pts

In a doubly-linked list, reverse traversal is possible with O(N) runtime complexity, where N is the number of items in the list.

Correct!

- [✓ True]
- False

Question 10

1 / 1 pts

Regarding insertion sort, which one of the list data structures is preferred considering runtime?

- ○ Singly-linked lists
- Correct! [✓ Doubly-linked lists]

Question 11

1 / 1 pts

What additional step is needed to perform in merge sort if singly-linked list is used rather than arrays?

- Correct! [✓ To identify the middle point, list needs to be traversed.]
- ○ Merging needs additonal space to store lists.

Question 12

1 / 1 pts

What is the benefit of using singly-linked lists in merge sort compared to arrays?

- ○ Less runtime complexity
- Correct! [✓ Merging does not need additonal container storage]

- ○ Constant time middle point detection

Question 13

1 / 1 pts

What is the problem of using singly-linked lists in quick sort?

- ○ Extra storage is required to perform partitioning.
- ○ Middle point is undetectable in partitioning singly-linked lists
- Correct! [✓ Partitioning needs to traverse the container in reverse, whereas singly-linked lists do not support it.]

Question 14

1 / 1 pts

Which one is preferred for shell sort?

- Correct! [✓ arrays]
- ○ singly-linked lists
- ○ doubly-linked lists

Quiz Score: **14** out of 14