Questions on LinkedList and ArrayList

1) What is included in a linked list node?

I a reference to the next node

II an array reference

III a data element

a) I

b) II

c) II and III

d) I and III

2) The Node class is a private inner class of the LinkedList class. Which of the following statements regarding this implementation is NOT correct?

a) The methods of the LinkedList class can access the public features of the Node class.

b) The methods of the Node class can access the public features of the LinkedList class.

c) The methods of the Node class can be directly accessed by other classes.

d) The Node class's instance variables that represent the node element and its next node reference are declared as public.

3) Insert the missing code in the following code fragment. This fragment is intended to add a new node to the head of a linked list:

public class LinkedList

{

. . .

public void addFirst(Object element)

{

Node newNode = new Node(); 1

newNode.data = element;

\_\_\_\_\_\_\_\_\_ 2

\_\_\_\_\_\_\_\_\_ 3

}

. . .

}

a)

first = newNode;

newNode.next = first;

b)

newNode.next = first;

first = newNode;

c)

first = newNode.next;

newNode.next = first;

d)

first = newNode.next;

newNode = first;

4) Insert the missing code in the following code fragment. This fragment is intended to remove a node from the head of a linked list:

public class LinkedList

{

. . .

public E removeFirst()

{

if (first == null) { throw new NoSuchElementException(); }

E element = first.data;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

}

. . .

}

a)

first = first.next; 1

return element;

b)

first.next = first; 1

return element;

c)

first = element.next; 1

return element;

d)

first = element.next; 1

return null;

5) Consider the following code snippet:

LinkedList<String> words = new LinkedList<String>();

words.addFirst("123");

words.addLast("456");

words.addFirst("789");

System.out.print(words.removeLast());

System.out.print(words.removeFirst());

System.out.print(words.removeLast());

What does this code print?

a) 123456789

b) 789123456

c) 123789456

d) 456789123

6) Consider the following code snippet:

LinkedList<String> words = new LinkedList<String>();

words.addFirst("xyz");

words.addLast("jkl");

words.addLast("def");

System.out.print(words.removeFirst());

System.out.print(words.removeLast());

System.out.print(words.removeLast());

What does this code print?

a) xyzjkldef

b) defxyzjkl

c) xyzdefjkl

d) defjklxyz

7) Adding or removing an element at an arbitrary iterator position in a singly linked list of length *n* takes \_\_\_\_ time.

a) *O*(*n*)

b) *O*(log *n*)

c) *O*(1)

d) *O*(*n*2)

8) If we want a create a doubly-linked list data structure so that we can move from a node to the next node as well as to a previous node, we need to add a previous reference to the Node class. Each such DNode (doubly-linked Node) will have a data portion and two DNode references, next and previous. How many references need to be updated when we remove a node from the middle of such a list? Consider the neighboring nodes.

a) 1

b) 2

c) 3

d) 4

9) If we want a create a doubly-linked list data structure so that we can move from a node to the next node as well as to a previous node we need to add a previous reference to the Node class. Each such DNode (doubly-linked Node) will have a data portion and two DNode references, next and previous. How many references need to be updated when we remove a node from the beginning of a list with many nodes? Consider the first reference and neighboring node(s).

a) 1

b) 2

c) 3

d) 4

10) In a linked list data structure, when does the reference to the first node need to be updated?

I inserting into an empty list

II deleting from a list with one node

III deleting an inner node

a) I

b) II

c) I and II

d) III

11) Suppose we maintain a linked list of length *n* in sorted order. What would be the big-Oh notation for the add operation?

a) *O*(1)

b) *O*(*n*)

c) *O*(*n* log2 *n*)

d) *O*(*n*2)

12) A doubly-linked list requires that each node maintain two references, one to the next node and one to the previous node. Which of the following statements about a doubly-linked list is NOT correct?

a) If a node's next reference is null, it is at the end of the list.

b) To remove a node in the middle of the list, the previous node's next reference must be updated.

c) To add a node in the middle of the list, you must update the next reference of the node after which the new node will be added.

d) To remove a node in the middle of the list, the previous node's previous reference must be updated.

13) Which of the following actions must be taken to remove a node X from the middle of a doubly-linked list?

I Update the next reference in the node before X

II Update the previous reference in the node after X

III Update the list's first reference

a) I

b) II

c) I and II

d) II and III

14) Which of the following actions must be taken to add a node X into the middle of a doubly-linked list?

I Update the next reference in the node before the position where X will be placed

II Update the previous reference in the node after the position where X will be placed

III Update the list's first reference

a) I

b) II

c) I and II

d) II and III

15) Which of the following actions must be taken to add a node X at the beginning of a doubly-linked list?

I Update the next reference in the node before the position where X will be placed

II Update the previous reference in the node after the position where X will be placed

III Update the list's first reference

a) I

b) II

c) I and II

d) II and III

16) Which of the following actions must be taken to add a node X at the end of a doubly-linked list?

I Update the next reference in the node before the position where X will be placed

II Update the previous reference in the node before the position where X will be placed

III Update the list's last reference

a) I

b) II

c) I and II

d) I and III

17) Reading or writing an array list element at an arbitrary index takes \_\_\_\_ time.

a) *O*(log (*n*))

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)

18) Adding or removing an arbitrary element in the middle of an array list takes \_\_\_\_ time.

a) *O*(*n*)

b) *O*(1)

c) *O*(log(*n*))

d) *O*(*n*2)

19) On average, how many elements of an array list of size *n* need to be moved when an element is removed?

a) *n*

b) *n2*

c) 2*n*

d) *n* / 2

20) On average, how many elements of an array list of size *n* need to be moved when an element is added?

a) *n*

b) *n2*

c) 2*n*

d) *n* / 2

21) Adding an element to the end of an array list, where the array list does not need to grow dynamically, takes \_\_\_\_ time.

a) *O*(*n*)

b) *O*(1)

c) *O*(log(*n*))

d) *O*(*n*2)

22) Which of the following statements about array list and doubly-linked list operations is correct?

a) It is more efficient to add an element in the middle of an array list than a doubly-linked list.

b) It is more efficient to add an element to the beginning of an array list than a doubly-linked list.

c) It is more efficient to remove an element in the middle of an array list than a doubly-linked list.

d) It is more efficient to retrieve an element in the middle of an array list than a doubly-linked list.

23) Array list operations that were studied included adding/removing an element at the end or in the middle, and retrieving the *k*th element. Which of the following statements about these array list operations is correct?

a) The most expensive operation of an array list is to add an element at the end.

b) The most expensive operation of an array list is to remove an element at the end.

c) The most expensive operation of an array list is to add an element in the middle.

d) The most expensive operation of an array list is to retrieve an arbitrary element.

24) You have implemented a queue as a singly-linked list, adding elements at the end and removing elements at the front. What is the cost of the add operation?

a) *O*(log *n*)

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)

25) You have implemented a queue as a singly-linked list, adding elements at the end and removing elements at the front. What is the cost of the remove operation?

a) *O*(log(*n*))

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)