Course Title: Big Java, Early Objects

Chapter Number: 16 Basic Data Structures

Question type: Multiple Choice

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

Answer: d

Title: What is included in a linked list node?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

2) In the textbook implementation, 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.

Answer: c

Title: Which statement regarding the implementation of the Node class is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

3) Which Java package contains the LinkedList class?

a) java.lang

b) java.util

c) java.collections

d) java.io

Answer: b

Title: Which Java package contains the LinkedList class?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

4) 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;

Answer: b

Title: Complete the code to add a node to the head of a linked list

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.2 Adding and Removing the First Element

5) 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 Object removeFirst()

{

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

Object 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;

Answer: a

Title: Complete the code to remove a node from the head of a linked list

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.2 Adding and Removing the First Element

6) 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 Object removeFirst()

{

if (first == null) { \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ }

Object element = first.data;

first = first.next; 1

return element;

}

. . .

}

a) throw new NoSuchElementException();

b) throw new IllegalStateException();

c) throw new NullPointerException();

d) throw new IllegalArgumentException();

Answer: a

Title: Complete the code to remove a node to the head of a linked list

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.2 Adding and Removing the First Element

7) Which of the following statements about removing a node from a linked list is correct?

a) A node's data is discarded when it is removed from the linked list, and its memory space is immediately reclaimed.

b) A node's data is returned to the program when the node is removed from the linked list, and its memory space is immediately reclaimed.

c) A node's data is discarded when it is removed from the linked list, and its memory space is reclaimed later by the garbage collector.

d) A node's data is returned to the program when the node is removed from the linked list, and its memory space is reclaimed later by the garbage collector.

Answer: d

Title: Which statement about removing a node from a linked list is correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.2 Adding and Removing the First Element

8) In the textbook implementation, the LinkedListIterator 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 LinkedListIterator class.

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

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

d) Clients of the LinkedListClass do not know the name of the LinkedListIterator class.

Answer: c

Title: Which statement regarding the implementation of the LinkedListIterator class is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.3 The Iterator Class

9) The linked list iterator described in the textbook maintains a reference to the last visited node, called position, and a reference to the last node before that, called previous. Which of the following statements is NOT correct regarding advancing this iterator?

a) If another node exists, when the iterator is to be advanced, the position reference must be updated to position.next.

b) If the iterator currently points before the first element of the list, when the iterator is to be advanced, position must be set to point to the first node in the linked list.

c) If another node exists, when the iterator is to be advanced, the previous reference must be updated to point to the current location of the iterator

d) If the iterator currently points before the first element of the list, when the iterator is to be advanced, the position reference must be set to position.next and the previous reference must be set to point to the first node in the linked list.

Answer: d

Title: Which statement about a linked list iterator is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.4 Advancing an Iterator

10) Which of the following statements about a linked list iterator is NOT correct?

a) The iterator is at the end of the list if the linked list's first node reference is null.

b) The iterator is at the end of the list if the position.next reference is null.

c) The iterator is at the end of the list if the position reference is null.

d) The list is empty if the linked list's first node reference is null.

Answer: c

Title: Which statement about a linked list iterator is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.4 Advancing an Iterator

11) Which of the following statements about a linked list and its iterator is NOT correct?

a) The list is empty if the linked list's first node reference is null.

b) The iterator is at the end of the list if the position.next reference is null.

c) The iterator is at the beginning of the list if the previous reference is null.

d) The iterator is at the first node of the list if its position reference is null.

Answer: d

Title: Which statement about a linked list iterator is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.4 Advancing an Iterator

12) Using the textbook's implementation of a singly linked list and linked list iterator, the following steps are required to remove a node from the middle of a linked list. Place these steps into the order in which they should be performed.

I The preceding node's next reference must be updated to skip the removed node.

II The iterator's position reference must be set to the previous reference.

III The previous reference must be checked to see if it is equal to the position reference.

a) III, I, II

b) I, III, II

c) II, I, III

d) III, II, I

Answer: a

Title: Order the steps for removing a node in the middle of a linked list.

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.5 Removing an Element

13) When using the textbook’s implementation of a singly linked list to remove an element in the middle of the list, why it is necessary to check whether the previous reference equals the position reference?

a) If previous equals position, the action does not follow a call to next.

b) If previous equals position and an attempt is made to remove the node, the iterator would have to start at the beginning of the list to rebuild the links.

c) If previous equals position, the iterator is at the beginning of the list and does not point to a valid node.

d) If previous equals position, the iterator is at the end of the list and does not point to a valid node.

Answer: a

Title: When removing a node from a linked list, why is it necessary to check if previous == position?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.5 Removing an Element

14) Using the textbook's implementation of a linked list, which of the following statements about adding a node to the middle of a linked list is correct?

a) The new node will be added before the last visited node.

b) The position.next reference will be updated to point to the new node.

c) The remove method can be called immediately before or after adding the new node.

d) The previous reference must be updated when adding the new node.

Answer: b

Title: Which statement about adding a new node in the middle of a linked list is correct?

Difficulty: Easy

Section Reference 1: 16.1.6 Adding an Element

15) 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

Answer: d

Title: What does this code print?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.6 Adding an Element

16) 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

Answer: c

Title: What does this code print?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.6 Adding an Element

17) Using the textbook's implementation of a linked list, which of the following statements about changing the data stored in a previously visited element is correct?

a) The node that will be updated is the most recently visited node.

b) The position.next reference must be updated when the data is updated.

c) The set method can be called again immediately after calling the add method.

d) The previous reference must be updated when the node is updated.

Answer: a

Title: Which statement about updating an existing node in a linked list is correct?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.7 Setting an Element to a Different Value

18) Using the textbook's implementation of a linked list, which of the following statements about managing nodes within a linked list using an iterator is correct?

a) The node that will be removed is the node pointed to by the position.next reference.

b) The set method can be called immediately after adding a new node using the add method.

c) The set method can be called immediately after removing an existing node using the remove method.

d) The position reference must be updated when a new node is added.

Answer: d

Title: Which statement about updating an existing node in a linked list is correct?

Difficulty: Easy

Section Reference 1: 16.1.7 Setting an Element to a Different Value

19) Assume that the linked list implementation includes a reference to the last node as well as to the first node. Which of the following statements about the efficiency of the linked list is correct?

a) Adding an element to the middle of the linked list at the current position of the iterator is *O*(*n*).

b) Removing an element other than the last element from the linked list at the current position of the iterator is *O*(*n*).

c) Accessing an element in the linked list using an iterator is *O*(*n*).

d) Adding an element to the end of the linked list is *O*(1).

Answer: d

Title: Which statement about the efficiency of a linked list is correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

20) Assume that the linked list implementation includes a reference to the last node as well as to the first node. Which of the following statements about the efficiency of a singly linked list is NOT correct?

a) Adding an element to the middle of a linked list using an iterator is *O*(1).

b) Removing the last element from a linked list using an iterator is *O*(1).

c) Accessing an element in a linked list using an iterator is *O*(*n*).

d) Adding an element to the end of a linked list is *O*(1).

Answer: b

Title: Which statement about the efficiency of a linked list is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

21) Which of the following operations is least efficient in a LinkedList?

a) adding an element in a position that has already been located

b) linear traversal step

c) removing an element when the element's position has already been located

d) random access of an element

Answer: d

Title: Which operation is least efficient in a LinkedList?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

22) Which of the following algorithms would be efficiently executed on a LinkedList?

a) tracking paths in a maze

b) binary search

c) remove first *n* / 2 elements from a list of *n* elements

d) read *n* / 2 elements in random order from a list of *n* elements

Answer: c

Title: Which algorithm would be efficiently executed on a LinkedList?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

23) What type of access does the use of an iterator with a LinkedList provide for its elements?

a) sequential

b) semi-random

c) random

d) sorted

Answer: a

Title: What type of access does a LinkedList iterator provide for list elements?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

24) 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)

Answer: c

Title: Adding or removing an element in a singly linked list of length *n* takes \_\_\_\_ time.

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

25) 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

Answer: b

Title: How many references need to be updated when removing a node from the middle of this doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

26) 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

Answer: b

Title: How many references need to update when removing a node from the beginning of this doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

27) 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

Answer: c

Title: In a linked list, when does the reference to the first node need to be updated?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

28) 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)

Answer: b

Title: What is the big-Oh notation for the add operation on this list?

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

29) Suppose we maintain two linked lists of length *n* in sorted order. What would be the big-Oh notation for the creating a third list, which included only elements common to both lists?

a) *O*(1)

b) *O*(*n*)

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

d) *O*(*n*2)

Answer: b

Title: What is big-Oh for creating a third list containing elements common to two sorted lists?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

30) Suppose we maintain two linked lists of length *n* in random element order. What would be the big-Oh notation for the creating a third list that includes only elements common to both lists, without sorting the first two lists?

a) *O*(1)

b) *O*(*n*)

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

d) *O*(*n*2)

Answer: d

Title: What is big-Oh for creating a third list containing elements common to two unsorted lists?

Difficulty: Hard

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

31) Suppose we maintain a linked list of length *n* in random element order. What would be the big-Oh notation for an algorithm that prints each list element and the number of times it occurs in the list (without sorting the list)?

a) *O*(1)

b) *O*(*n*)

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

d) *O*(*n*2)

Answer: d

Title: What is big-Oh for printing out each list element plus its number of occurrences?

Difficulty: Hard

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

32) Suppose we maintain a linked list of length *n* in random element order. What would be the big-Oh notation for printing out those elements which occur exactly once in the list (without sorting the list)?

a) *O*(1)

b) *O*(*n*)

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

d) *O*(*n*2)

Answer: d

Title: What is big-Oh for printing out list elements that occur exactly once in the list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

33) Suppose we maintain a linked list of length *n* in sorted order. What would be the big-Oh notation for printing out those elements that occur exactly once in the list?

a) *O*(1)

b) *O*(*n*)

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

d) *O*(*n*2)

Answer: b

Title: What is big-Oh for printing those elements that occur exactly once in a sorted list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

34) 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.

Answer: d

Title: Which statement about a doubly-linked list is NOT correct?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

35) 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

Answer: c

Title: Which actions must be taken to remove a node X from the middle of a doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

36) 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

Answer: c

Title: Which actions must be taken to add a node X into the middle of a doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

37) 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

Answer: d

Title: Which actions must be taken to add a node X at the beginning of a doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

38) 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

Answer: d

Title: Which actions must be taken to add a node X at the end of a doubly-linked list?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: 16.1.8 Efficiency of Linked List Operations

39) Using the textbook's implementation of a linked list, what is the purpose of declaring the Node class to be a static inner class?

a) To create an outer-class reference.

b) To create a this reference to itself.

c) To prevent storage of an outer-class reference that is not needed.

d) To create an outer-class reference that is needed.

Answer: c

Title: What is the purpose of making the Node class a static inner class?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: Special Topic 16.1 Static Classes

40) Using the textbook's implementation of a linked list, why is the LinkedListIterator inner class NOT declared as an inner static class?

a) Because the LinkedList class must have access to the instance variables of the LinkedListIterator class.

b) Because the Node class must have access to the instance variables of the LinkedListIterator class.

c) Because the LinkedListIterator class must have access to Node class instance variables.

d) Because the LinkedListIterator class must have access to LinkedList class instance variables.

Answer: d

Title: Why is the LinkedListIterator inner class NOT a static class?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 2: Special Topic 16.1 Static Classes

41) What is never present in a static inner class?

a) an outer-class reference.

b) the this reference to itself.

c) static properties.

d) static methods.

Answer: a

Title: What is never present in a static inner class?

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

Section Reference 1: Special Topic 16.1 Static Classes

42) Given the partial LinkedList class declaration below, select a statement to complete the printFirst method, which is designed to display the contents of the first list element.

public class LinkedList

{

class Node

{

public Object data;

public Node next;

}

private Node first;

. . .

public void printFirst()

{

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

}

}

a) System.out.println(first);

b) System.out.println(first.data);

c) System.out.println(first.next);

d) System.out.println(Node.data);

Answer: b

Title: Select statement to display the contents of the first element in a linked list

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

43) Given the partial LinkedList class declaration below, select an expression to complete the empty method, which is designed to return true if the list contains no elements.

public class LinkedList

{

class Node

{

public Object data;

public Node next;

}

private Node first;

. . .

public boolean empty()

{

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

}

}

a) first != null

b) first == null

c) first.data == null

d) first.next == null

Answer: b

Title: Select statement to complete the Linked List empty method

Difficulty: Easy

Section Reference 1: 16.1 Implementing Linked Lists

44) Given the partial LinkedList class declaration below, select a statement to complete the size method, which is designed to return the number of list elements.

public class LinkedList

{

class Node

{

public Object data;

public Node next;

}

private Node first;

. . .

public int size()

{

int count = 0;

Node temp = first;

while (temp != null)

{

count++;

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

}

return count;

}

}

a) temp = temp.next;

b) temp = first.next;

c) first = temp.next;

d) first = first.next;

Answer: a

Title: Select statement to complete linked list size method

Difficulty: Medium

Section Reference 1: 16.1 Implementing Linked Lists

45) Given the partial LinkedList and LinkedListIterator class declarations below, select an expression to complete the LinkedList get(index) method, which returns the element at the position indicated by index.

public class LinkedList

{

. . .

public ListIterator listIterator()

{

return new LinkedListIterator();

}

class LinkedListIterator implements ListIterator

{

private Node position;

private Node previous;

private boolean isAfterNext;

public LinkedListIterator()

{

. . .

}

public Object next()

{

. . .

}

public boolean hasNext()

{

. . .

}

}

public Object get(int index)

{

ListIterator it = listIterator();

for (int i = 0; i < index; ++i)

{

it.next();

}

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

}

}

a) it.next()

b) it.previous

c) it.position

d) it.next().data

Answer: a

Title: Select expression to complete LinkedList get(index) method

Difficulty: Hard

Section Reference 1: 16.1 Implementing Linked Lists

46) Given the partial ArrayList class declaration below, select an expression to complete the empty method, which is designed to return true if the list contains no elements.

public class ArrayList

{

private Object[] elements;

private int currentSize;

public ArrayList()

{

final int INITIAL\_SIZE = 10;

elements = new Object[INITIAL\_SIZE];

currentSize = 0;

}

public boolean empty()

{

return \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ;

}

}

a) elements.length == 0

b) elements.currentSize == 0

c) elements[0] == null

d) currentSize == 0

Answer: d

Title: Select statement to complete ArrayList empty method

Difficulty: Easy

Section Reference 1: 16.2 Implementing Array Lists

47) Given the partial ArrayList class declaration below, select an expression to complete the contains method.

public class ArrayList

{

private Object[] elements;

private int currentSize;

public ArrayList()

{

final int INITIAL\_SIZE = 10;

elements = new Object[INITIAL\_SIZE];

currentSize = 0;

}

public boolean contains(Object item)

{

for (int i = 0; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ; i++)

{

if (elements[i].equals(item))

{

return true;

}

}

return false;

}

...

}

a) i < currentSize

b) i <= currentSize

c) i < elements.length

d) i <= elements.length

Answer: a

Title: Select expresson to complete ArrayList contains method

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

48) An array list maintains a reference to an array of elements called a \_\_\_\_.

a) buffer

b) tree map

c) hash table

d) bucket

Answer: a

Title: An array list maintains a reference to an array of elements called a/an \_\_\_\_.

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

49) 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)

Answer: d

Title: Random access in an array list takes \_\_\_\_ time.

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

50) Suppose we maintain an array A of *n* int values as follows:

A[0] < A[1] < . . . < A[i] > A[i + 1] > A[i + 2] > . . . > A[n - 1]

The *i*th element is the maximum in the array. What would be the lowest big-Oh notation for finding that element? Consider a variation of the binary search.

a) *O*(1)

b) *O*(log2 *n*)

c) *O*(*n*)

d) *O*(*n*2)

Answer: b

Title: What is the lowest big-Oh notation for finding the maximum in this array?

Difficulty: Hard

Section Reference 1: 16.2 Implementing Array Lists

51) What feature of the ArrayList class makes it much better for a binary search than the LinkedList class?

a) indexing

b) has no link references

c) it is a generic class

d) it is an abstract class

Answer: a

Title: What feature of ArrayList makes it better for a binary search than LinkedList?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

52) Array lists and linked lists both have the same \_\_\_\_.

a) add/remove efficiency

b) concrete implementations

c) random element access efficiency

d) linear traversal step efficiency

Answer: d

Title: Array lists and linked lists both have the same \_\_\_\_.

Difficulty: Easy

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.2 Removing or Adding Elements

53) 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)

Answer: a

Title: Adding or removing an arbitrary element in an array list takes \_\_\_\_ time.

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.2 Removing or Adding Elements

54) 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

Answer: d

Title: On average, how many elements of an array list need to move when an element is removed?

Difficulty: Easy

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.2 Removing or Adding Elements

55) 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

Answer: d

Title: On average, how many elements of an array list need to move when an element is added?

Difficulty: Easy

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.2 Removing or Adding Elements

56) If the current size of an array list is less than the length of the buffer, adding an element at the end of an array list takes \_\_\_\_ time.

a) *O*(*n*)

b) *O*(1)

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

d) *O*(*n*2)

Answer: b

Title: If the array list size is less than the buffer’s, adding an element at the end takes \_\_\_\_ time.

Difficulty: Easy

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.2 Removing or Adding Elements

57) When the buffer for an array list must be grown, a single reallocation operation takes \_\_\_\_ time.

a) *O*(*n*)

b) *O*(1)

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

d) *O*(*n*2)

Answer: a

Title: When the buffer for an array list must be grown, reallocation is an \_\_\_\_ operation.

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

58) When considering the reallocation operation for a list whose buffer is full, on average it will take \_\_\_\_ time.

a) *O*(*n*)

b) *O*(1)

c) *O*(1)*+*

d) *O*(*n*2)

Answer: c

Title: When considering the reallocation operation for a list whose buffer is full, on average it will take \_\_\_\_ time.

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

59) 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.

Answer: d

Title: Which statement about array list and doubly-linked list operations is correct?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

60) 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.

Answer: c

Title: Which statement about efficiency of array list operations is correct?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

61) 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 least expensive operation of an array list is to add an element at the end.

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

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

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

Answer: d

Title: Which statement about efficiency of array list operations is correct?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

62) Linked list operations that were studied included adding/removing an element at the end or in the middle, and retrieving the *k*th element. If the iterator is currently pointing to the correct location for insertion or removal, which of the following statements about these doubly-linked list operations is correct?

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

b) The most expensive operation of a doubly-linked list is to remove an element at the end.

c) The most expensive operation of a doubly-linked list is to add an element in the middle.

d) The most expensive operation of a doubly-linked list is to retrieve an arbitrary element.

Answer: d

Title: Which statement about efficiency of doubly-linked list operations is correct?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

63) Linked list operations that were studied included adding/removing an element at the end or in the middle, and retrieving the *k*th element. If the iterator is currently pointing to the correct location for insertion or removal, which of the following statements about these doubly-linked list operations is correct?

a) The least expensive operation of a doubly-linked list is to add an element at the end.

b) The least expensive operation of a doubly-linked list is to retrieve an arbitrary element.

c) The least expensive operation of a doubly-linked list is to add an element in the middle.

d) All of these operations have the same time cost.

Answer: c

Title: Which statement about efficiency of doubly-linked list operations is correct?

Difficulty: Medium

Section Reference 1: 16.2 Implementing Array Lists

Section Reference 2: 16.2.3 Growing the Buffer

64) Which operations from the array list data structure could be used in the implementation of the push and pop operations of a stack data structure?

I addLast

II addFirst

III removeFirst

a) I

b) II

c) I and II

d) II and III

Answer: d

Title: Which array list operations could be used to implement push and pop operations?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

65) Which of the following operations from the array list data structure could be used in the implementation of the push and pop operations of a stack data structure?

I addLast

II addFirst

III removeLast

a) I

b) II

c) I and III

d) II and III

Answer: b

Title: Which array list operations could be used to implement push and pop operations?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

66) Complete the following code, which is intended to add an element to the top of a stack implemented as a linked list.

Node newNode = new Node();

newNode.data = element;

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

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

a)

first = newNode;

newNode.next = first;

b)

newNode.next = first;

first = newNode;

c)

newNode.previous = first;

first.next = newNode;

d)

first = newNode;

newNode.previous = first;

Answer: b

Title: Complete the code to add an element to the top of a stack implemented as a linked list.

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

67) A stack can be implemented as a sequence of nodes in a linked list or an array list. Which of the following statements about this is correct?

a) If implementing the stack as a linked list, the least expensive approach is to add and remove elements at the end.

b) If implementing the stack as an array list, the least expensive approach is to add and remove elements at the end.

c) If implementing the stack as an array list, there is no cost difference whether adding and removing elements at the beginning or the end.

d) If implementing the stack as a linked list, there is no cost difference whether adding and removing elements at the beginning or the end.

Answer: b

Title: Which statement about implementing stacks as linked lists or array lists is correct?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

68) A stack can be implemented as a sequence of nodes in a linked list or an array list. Which of the following statements about this is correct?

a) If implementing the stack as a linked list, it is more expensive to add and remove elements at the end than at the beginning.

b) If implementing the stack as an array list, it is more expensive to add and remove elements at the end than at the beginning.

c) If implementing the stack as an array list, there is no cost difference whether adding and removing elements at the beginning or the end.

d) If implementing the stack as a linked list, there is no cost difference whether adding and removing elements at the beginning or the end.

Answer: a

Title: Which statement about implementing stacks as linked lists or array lists is correct?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

Section Reference 2: 16.3.2 Stacks as Arrays

69) When implementing a queue as a singly-linked list, which of these statements is correct?

a) For better efficiency, nodes should be added at the back and removed at the front.

b) For better efficiency, nodes should be added at the front and removed at the back.

c) There is no difference in efficiency whether nodes are added at the front and removed at the back, or added at the back and removed at the front.

d) You cannot effectively implement a queue as a singly-linked list.

Answer: a

Title: Which statement about implementing queues as linked lists is correct?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

Section Reference 1: 16.3.3 Queues as Linked Lists

70) 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)

Answer: d

Title: What is the cost of the add operation on a queue implemented as a singly-linked list?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

Section Reference 2: 16.3.3 Queues as Linked Lists

71) 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)

Answer: d

Title: What is the cost of the remove operation on a queue implemented as a singly-linked list?

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

Section Reference 2: 16.3.3 Queues as Linked Lists

72) Given the ArrayStack class implementation discussed in section 16.3 (partially shown below), select the statements needed to complete the push method.

public class ArrayStack

{

private Object[] elements;

private int currentSize;

public ArrayStack()

{

final int INITIAL\_SIZE = 10;

elements = new Object[INITIAL\_SIZE];

currentSize = 0;

}

public void push(Object element)

{

growIfNecessary();

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

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

}

}

a)

elements[currentSize] = element;

currentSize++;

b)

currentSize++;

elements[currentSize] = element;

c)

elements[currentSize - 1] = element;

currentSize++;

d)

elements[currentSize + 1] = element;

currentSize++;

Answer: a

Title: Select statements to complete push method for ArrayStack class

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

73) Given the LinkedListStack class implementation discussed in section 16.3 (partially shown below), select the statement(s) to complete the peek method.

public class LinkedListStack

{

private Node first;

public LinkedListStack()

{

first = null;

}

public Object peek()

{

if (empty())

{

throw new NoSuchElementException();

}

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

}

...

}

a)

Object value = first.data;

first = first.next;

return value;

b)

first = first.next;

return first.data;

c)

return first;

d)

return first.data;

Answer: d

Title: Select statement(s) to complete peek method for LinkedListStack class

Difficulty: Medium

Section Reference 1: 16.3 Implementing Stacks and Queues

74) Given the LinkedListQueue class implementation discussed in section 16.3 (partially shown below), select the appropriate statements to complete the add method.

public class LinkedListQueue

{

private Node first;

private Node last;

public LinkedListQueue()

{

first = null;

last = null;

}

public void add(Object newElement)

{

Node newNode = new Node();

newNode.data = newElement;

newNode.next = null;

if (last == null)

{

first = newNode;

last = newNode;

}

else

{

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

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

}

}

...

}

a)

last.next = newNode.next;

last = newNode;

b)

last.next = newNode;

last = newNode;

c)

last = newNode;

last.next = newNode;

d)

last = newNode;

last.next = newNode.next;

Answer: b

Title: Select statements to complete add method for LinkedListQueue class

Difficulty: Hard

Section Reference 1: 16.3 Implementing Stacks and Queues

75) Given the HashSet class implementation discussed in section 16.4 (partially shown below), select the statement needed to complete the clear method, which is designed to remove all elements from the set.

public class HashSet

{

private Node[] buckets;

private int currentSize;

public HashSet(int bucketsLength)

{

buckets = new Node[bucketsLength];

currentSize = 0;

}

public void clear()

{

for (int j = 0; j < buckets.length; ++j)

{

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

}

currentSize = 0;

}

...

}

a) buckets[j] = 0;

b) buckets[j] = new Node();

c) buckets[j] = null;

d) buckets[j] = new LinkedList();

Answer: c

Title: Select statement to complete clear method for HashSet class

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

76) Elements in a hash table are said to \_\_\_\_ when they have the same hash code value.

a) be equivalent

b) compress

c) collide

d) buffer well

Answer: c

Title: Elements in a hash table are said to \_\_\_\_ when they have the same hash code value.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

77) A hash function is considered good if it \_\_\_\_.

a) does not require compression.

b) detects duplicate elements.

c) results in low integer values.

d) minimizes collisions.

Answer: d

Title: A hash function is considered good if it \_\_\_\_.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

78) Which of the following statements about hash tables is correct?

a) The hash code is used to determine where to store each element.

b) Elements in the hash table are sorted in hash code order.

c) A hash table allows duplicate elements.

d) No two elements of a hash table can have the same hash code.

Answer: a

Title: Which statement about hash tables is correct?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

79) Which of the following statements about hash tables is NOT correct?

a) Each entry in a hash table points to a sequence of nodes whose elements have the same compressed hash code.

b) Elements with the same hash code are stored as nodes in a bucket associated with that hash code.

c) A compressed hash code is used as an array index into the hash table.

d) All elements of a hash table must be searched sequentially to determine if an element already exists.

Answer: d

Title: Which statement about hash tables is NOT correct?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

80) Assume that you have a hash table in which there are few or no collisions. What is the time required to locate an element in this hash table?

a) *O*(log *n*)

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)

Answer: d

Title: What is the time required to locate an element in a hash table with few collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

81) In the separate chaining technique for handling collisions in a hash table, \_\_\_\_.

a) colliding elements are stored in a nested hash table.

b) colliding elements are placed in empty locations in the hash table.

c) colliding elements are stored in linked lists associated with the hash code.

d) the hash code is compressed to obtain unique hash codes.

Answer: c

Title: In the separate chaining technique for handling collisions in a hash table, \_\_\_\_.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

82) In the open addressing technique for handling collisions in a hash table, \_\_\_\_.

a) colliding elements are stored in a nested hash table.

b) colliding elements are placed in empty locations in the hash table.

c) colliding elements are stored in linked lists associated with the hash code.

d) the hash code is compressed to obtain unique hash codes.

Answer: b

Title: In the open addressing technique for handling collisions in a hash table, \_\_\_\_.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

83) Complete the following code snippet, which is intended to compress a hash code to become a valid array index:

int h = x.hashCode();

if (h < 0) { h = -h; }

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

a) position = arrayLength % h;

b) position = arrayLength / h;

c) position = h / arrayLength;

d) position = h % arrayLength;

Answer: d

Title: Complete the code to compress a hash code.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

84) Complete the following code snippet, which is intended to compress a hash code to become a valid array index:

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

if (h < 0) { h = -h; }

position = h % arrayLength;

a) double h = x.hashCode();

b) double h = x.getHashCode();

c) int h = x.hashCode();

d) int h = x.getHashCode();

Answer: c

Title: Complete the code to compress a hash code.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

85) Why is it not typical to use the hashCode method result directly as an index for array storage?

I because the hashcode method returns a double

II the values are potentially very large

III the values are not type int

a) I

b) I and II

c) I and III

d) II

Answer: d

Title: Why is it not typical to use the hashCode directly as an index for array storage?

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

86) Consider the following code snippet, which computes h, the array index for storing x in a hash table.

int h = x.hashCode();

if (h < 0) { h = -h; }

h = h % size;

What does size correspond to?

a) The size of the hash table.

b) The number of elements to be stored.

c) The number of collisions.

d) The index of an empty hash table slot.

Answer: a

Title: What does size correspond to in this code snippet for hash tables?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

87) What technique is used to store elements that hash to the same location?

a) colliding

b) bucketing

c) deletion

d) sharing

Answer: b

Title: What technique is used to store elements that hash to the same location?

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.2 Hash Tables

88) Assume that you have a hash table in which there are few or no collisions. What is the time required to add a new element to this hash table?

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

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)

Answer: d

Title: What is the time required to add a new element to a hash table with few collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.3 Finding an Element

89) Assume that you have a hash table in which there are few or no collisions. What is the time required to remove an element from this hash table?

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(1)

d) *O*(log (*n*))

Answer: c

Title: What is the time required to remove an element from this hash table with few collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.3 Finding an Element

90) Which of the following statements about adding an element to a hash table is NOT correct?

a) Add the new element at the beginning of the node sequence in the bucket referenced by the hash code.

b) Check the elements in the bucket to determine if the new element already exists.

c) If the element matches another element in the bucket referenced by the hash code, add the new element to that bucket.

d) To add an element, its compressed hash code must be computed.

Answer: c

Title: Which statement about adding an element to a hash table is NOT correct?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.4 Adding and Removing Elements

91) If your hashCode function returns a number anywhere in the hash table with equal probability, what is the likely result?

a) Some objects will be impossible to find.

b) The number of collisions will be high.

c) The get method will run at *O*(*n*) complexity.

d) The get method will run at *O*(1) complexity.

Answer: d

Title: If your hashCode function returns a number anywhere in the hash table with equal probability, what is the likely result?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.4 Adding and Removing Elements

92) Which hash table method(s) will make use of the equals method?

I put

II get

III contains

a) I

b) I and II

c) III

d) I, II and III

Answer: d

Title: Which hash table method(s) will make use of the equals method?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

93) Which of the following statements about using iterators with hash tables is NOT correct?

a) The iterator must track its position within a node chain in a bucket.

b) The iterator must skip over empty buckets.

c) Two iterators are required: one to traverse the buckets, and another to traverse the nodes within a bucket.

d) The iterator must track the bucket number.

Answer: c

Title: Which statement about using iterators with hash tables is NOT correct?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

94) What is the time required to iterate over all elements in a hash table of size *n*?

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

b) *O*(*n*)

c) *O*(*n*2)

d) *O*(1)

Answer: b

Title: What is the time required to iterate over all elements in a hash table of size *n*?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

95) Assume that you have a hash table in which there are an average number of collisions. What is the time required to remove an element from this hash table?

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(1)

d) *O*(1)*+*

Answer: d

Title: What is the time required to remove an element from a hash table with average collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

96) Assume that you have a hash table in which there are an average number of collisions. What is the time required to find an element in this hash table?

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(1)

d) *O*(1)*+*

Answer: c

Title: What is the time required to find an element in a hash table with average collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

97) Assume that you have a hash table in which there are an average number of collisions. What is the time required to add an element to this hash table?

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(1)

d) *O*(1)*+*

Answer: d

Title: What is the time required to add an element to this hash table with average collisions?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

98) Complete the following code, which is intended to add an element to a hash table. Assume that the computed and compressed hash code is stored in the variable h.

Node newNode = new Node();

newNode.data = x;

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

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

a)

newNode.next = buckets[h + 1];

buckets[h] = newNode;

b)

newNode.next = buckets[h];

buckets[h + 1] = newNode;

c)

newNode.next = buckets[h];

buckets[h - 1] = newNode;

d)

newNode.next = buckets[h];

buckets[h] = newNode;

Answer: d

Title: Complete the code to add an element to a hash table.

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

99) What type of access does the use of an iterator provide for the elements of a bucket in a hash table?

a) sequential

b) semi-random

c) random

d) sorted

Answer: a

Title: What type of access does the use of an iterator provide for the elements of a bucket in a hash table?

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: 16.4.5 Iterating over a Hash Table

100) The advantage of using the open addressing technique over the separate chaining technique for handling collisions in a hash table is that open addressing \_\_\_\_.

a) allows for faster addition of elements

b) allows for faster retrieval of elements

c) is simpler in implementation

d) requires less memory storage for the hash table

Answer: d

Title: The advantage of using open addressing over separate chaining is \_\_\_\_.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: Special Topic 16.2 Open Addressing

101) The \_\_\_ technique for handling collisions in a hash table with open addressing attempts to place colliding elements at the next available location in the hash table.

a) sequential placement

b) sequential probing

c) linear placement

d) linear probing

Answer: d

Title: The \_\_\_ technique attempts to place colliding elements at the next available location.

Difficulty: Easy

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: Special Topic 16.2 Open Addressing

102) Which statement about handling collisions in a hash table using the open addressing technique is correct?

a) A colliding element will always be contiguous to the location in which it would normally be placed.

b) To find an element, you must search from the hash code location until a match or an element with a different hash code is found.

c) To remove an element, you simply empty the slot at which you find it.

d) There may be some elements with different hash codes that lie on the same probing sequence.

Answer: d

Title: Which statement about handling collisions using open addressing is correct?

Difficulty: Hard

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: Special Topic 16.2 Open Addressing

103) Which of the following statements about handling collisions in a hash table using the sequential chaining and open addressing techniques is NOT correct?

a) The implementation of sequential chaining technique is simpler than the implementation of the open addressing technique.

b) The sequential chaining technique is simpler to understand than the open addressing technique.

c) The sequential chaining technique requires the storage of links while open addressing does not.

d) The sequential chaining technique requires less memory use than open addressing.

Answer: d

Title: Which statement about sequential chaining and open addressing is NOT correct?

Difficulty: Medium

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: Special Topic 16.2 Open Addressing

104) Which statement about handling collisions in a hash table using the open addressing technique is NOT correct?

a) A colliding element may not be contiguous to the location in which it would normally be placed.

b) To find an element, you must search from the hash code location until a match or an empty slot is found.

c) To remove an element, you simply empty the slot at which you find it.

d) There may be some elements with different hash codes that lie on the same probing sequence.

Answer: c

Title: Which statement about handling collisions using open addressing is NOT correct?

Difficulty: Hard

Section Reference 1: 16.4 Implementing a Hash Table

Section Reference 2: Special Topic 16.2 Open Addressing