Course Title: Big Java, Early Objects

Chapter Number: 14 Sorting and Searching

Question type: Multiple Choice

1) What type of algorithm places elements in order?

a) sorting

b) searching

c) insertion

d) deletion

Answer: a

Title: What type of algorithm places elements in order?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

2) In each iteration, selection sort places which element in the correct location?

a) The smallest in the array.

b) The smallest element not yet placed in prior iterations.

c) The largest element in the array.

d) A random element.

Answer: b

Title: In each iteration, selection sort places which element in the correct location?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

3) After 5 iterations of selection sort working on an array of 10 elements, what must hold true?

a) 5 more iterations are always necessary to complete the sort

b) 4 more iterations are always necessary to complete the sort

c) 5 more iterations may be needed to complete the sort

d) 4 more iterations may be needed to complete the sort

Answer: b

Title: After 5 iterations of selection sort on an array, what must hold true?

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

4) After one iteration of selection sort working on an array of 10 elements, what must hold true?

a) The array cannot be sorted.

b) One element must be correctly placed.

c) At least two elements are correctly placed.

d) The largest element is correctly placed.

Answer: b

Title: After one iteration of selection sort on an array, what must hold true?

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

5) Consider the sort method shown below for selection sort:

public static void sort(int[] a)

{

for (int i = 0; i < a.length – 1; i++)

{

int minPos = minimumPosition(i);

swap(minPos, i);

}

}

Suppose we modify the loop condition to read i < a.length. What would be the result?

a) An exception would occur.

b) The sort would work, but run one more iteration.

c) The sort would work but with one less iteration.

d) The sort would work exactly the same as before the code modification.

Answer: b

Title: What would happen if we modify the selection sort sort method loop condition?

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

6) Consider the sort method shown below for selection sort:

public static void sort (int[] a)

{

for (int i = 0; i < a.length – 1; i++)

{

int minPos = minimumPosition(i);

swap(minPos, i);

}

}

Suppose we modify the call to the swap method call to read swap(i, minPos). What would be the result?

a) An exception would occur.

b) The sort would produce incorrect results.

c) The sort would work, but sort backwards.

d) The sort would produce correct results.

Answer: d

Title: What would result from this call to the swap method: swap(i, minPos)?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

7) Consider the sort method for selection sort shown below:

public static void sort (int[] a)

{

for (int i = 0; i < a.length – 1; i++)

{

int minPos = minimumPosition(i);

swap(minPos, i);

}

}

Suppose we modify the loop control to read int i = 1; i < a.length – 1; i++. What would be the result?

a) An exception would occur

b) The sort would not consider the last array element.

c) The sort would not consider the first array element.

d) The sort would still work correctly.

Answer: c

Title: Consider the sort method for selection sort. What would be the result from this code change?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

8) Consider the minimumPosition method from the SelectionSorter class. Complete the code to write a maximumPosition method that returns the index of the largest element in the range from index from to the end of the array.

private static int minimumPosition(int[] a, int from)

{

int minPos = from;

for (int i = from + 1; i < a.length; i++)

{

if (a[i] < a[minPos]) { minPos = i; }

}

return minPos;

}

private static int maximumPosition(int[] a, int from)

{

int maxPos = from;

for (int i = from + 1; i < a.length; i++)

{

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

}

return maxPos;

}

a) if(a[i] > a[maxPos]) { maxPos = i; }

b) if(a[i] == a[maxPos]) { maxPos = i; }

c) if(a[i] < a[maxPos]) { maxPos = i; }

d) if(a[i] <= a[maxPos]) { maxPos = i; }

Answer: a

Title: Complete the code to return the index of the largest element in a range in an array.

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

9) Consider the swap method shown below from the SelectionSorter class. If we modified it as shown in the swap2 method shown below, what would be the effect on the sort method?

private static void swap(int[] a, int i, int j)

{

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

private static void swap2(int[] a, int i, int j)

{

a[i] = a[j];

a[j] = a[i];

}

a) There would be no effect.

b) Some array elements would be overwritten.

c) It would sort the array in reverse order.

d) It would still be correct, but run a little faster.

Answer: b

Title: If we modified SelectionSorter.swap, what would be the effect on the sort method?

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

10) Suppose you wanted to test your sort on an array filled with different elements each time the code is run. What is an efficient technique for creating an array of 1,000 elements for each run?

a) Run the program many times, entering different values for the array elements.

b) Make a file with many sets of values and loop through them, sorting each one.

c) Use the Random class to generate array elements, sorting each set in a loop.

d) Create many different arrays with different elements in the program code and sort each array.

Answer: c

Title: What is an efficient technique to create 1,000 elements for each sort test run?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

11) After 9 iterations of selection sort working on an array of 10 elements, what must hold true?

a) The largest element is correctly placed by default.

b) One more iteration is needed to complete the sort.

c) The smallest element is incorrectly placed.

d) The largest element is incorrectly placed.

Answer: a

Title: After 9 iterations of selection sort on an array of 10 elements, what must hold true?

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

12) Which selection sort iteration guarantees the array is sorted for a 10-element array?

a) 9th iteration

b) 10th iteration

c) 1st iteration

d) impossible to tell

Answer: a

Title: Which selection sort iteration guarantees the array is sorted for a 10-element array?

Difficulty: Easy

Section Reference 1: 14.1 Selection Sort

13) The largestPosition method below returns the index of the largest element in the tail range of an array of integers. Select the expression that would be needed to complete the selectionSort method below, so that it sorts the elements in descending order.

/\*\*

Finds the largest element in the tail range of an array.

@param a the array to be searched

@param from the first position in a to compare

@return the position of the largest element in range a[from]..a[a.length – 1]

\*/

private static int largestPosition(int[] a, int from)

{

int maxPos = from;

for (int j = from + 1; j < a.length; j++)

{

if (a[j] > a[maxPos])

{

maxPos = j;

}

}

return maxPos;

}

public static void selectionSort(int[] a)

{

for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

int maxPos = largestPosition(a, i);

ArrayUtil.swap(a, maxPos, i);

}

}

a) (int i = 0; i < a.length – 1; i++)

b) (int i = 0; i < a.length; i++)

c) (int i = a.length; i > 0; i--)

d) (int i = a.length – 1; i > 0; i--)

Answer: a

Title: Complete alternative selection sort algorithm to sort array in descending order

Difficulty: Medium

Section Reference 1: 14.1 Selection Sort

14) The code segment below is designed to add the elements in an array of integers. Select the expression needed to complete the code segment so that it calculates the running time elapsed.

long start = System.currentTimeMillis();

int sum = 0;

for (int k = 0; k < values.length; k++)

{

sum = sum + values[k];

}

long runningTime = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

a) System.currentTimeMillis()

b) System.currentTimeMillis() - start

c) System.currentTimeMillis() + start

d) runningTime + start - System.currentTimeMillis()

Answer: b

Title: Complete code segment to measure running time elapsed

Difficulty: Easy

Section Reference 1: 14.2 Profiling the Selection Sort Algorithm

15) The performance of an algorithm is most closely related to what?

a) The total number of element visits

b) The total number of elements

c) The type of elements

d) The number of lines of code in the method

Answer: a

Title: The performance of an algorithm is related to what?

Difficulty: Easy

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

16) Consider an array with *n* elements. If we visit each element *n* times, how many total visits will there be?

a) *n*

b) 2*n*

c) *nn*

d) *n*2

Answer: d

Title: If we visit each of *n* elements *n* times, how many total visits will there be?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

17) Suppose an array has *n* elements. We visit element #1 one time, element #2 two times, element #3 three times, and so forth. How many total visits will there be?

a) 2*n*

b) *n*(*n*+1)/2

c) *n*2

d) *n*3

Answer: b

Title: How many total element visits will there be in this scenario?

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

18) Suppose an algorithm requires a total of 3*n*3 + 2*n*2 – 3*n* + 4 visits. In big-Oh notation, the total number of visits is of what order?

a) *n*2 \* *n*2

b) *n*2

c) *n*6

d) *n*3

Answer: d

Title: In big-Oh notation, the total number of visits for this algorithm is of what order?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

19) In big-Oh notation, when we consider the order of the number of visits an algorithm makes, what do we ignore?

I power of two terms

II the coefficients of the terms

III all lower order terms

a) I

b) II

c) I and II

d) II and III

Answer: d

Title: What do we ignore when we consider the order of the number of visits?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

20) In big-Oh notation, suppose an algorithm requires an order of *n*3 element visits. How does doubling the number of elements affect the number of visits?

a) It doubles the number of visits.

b) It quadruples the number of visits.

c) It triples the number of visits.

d) It number of visits goes up by a factor of eight.

Answer: d

Title: How does doubling the number of elements affect the number of visits?

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

21) What is the smallest value of *n* for which *n*2 > 3*n* + 4?

a) 6

b) 5

c) 4

d) 3

Answer: b

Title: What is the smallest value of n for which *n*2 > 3n + 4?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

22) How large does *n* need to be so 0.3*n*2 is greater than 2.5*n* – 3?

a) 3

b) 5

c) 7

d) 9

Answer: c

Title: How large does *n* need to be so 0.3*n*2 is greater than 2.5*n* – 3?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

23) Which of the following completes the selection sort method minimumPosition()?

private static int minimumPosition(int[] a, int from)

{

int minPos = from;

for (int i = from + 1; i < a.length; i++)

{

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

}

return minPos;

}

a) if (a[i] > a[minPos]) { minPos = i; }

b) if (a[i] < a[minPos]) { minPos = i; }

c) if (a[i] < a[j]) { minPos = i; }

d) if (a[i] < a[minPos]) { i = minPos; }

Answer: b

Title: Which statement completes selection sort method minimumPosition()?

Difficulty: Easy

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

24) If you increase the size of a dataset fivefold, how much longer does it take to sort it with the selection sort algorithm?

a) Approximately 5 times longer

b) Approximately 20 times longer

c) Approximately 25 times longer

d) Approximately 100 times longer

Answer: c

Title: If a dataset increases fivefold, how much longer does selection sort take?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

25) How many comparisons does selection sort make when sorting an array of length *n*?

a) *n*

b) log(2*n*)

c) *n*( *n* + 1) / 2

d) *n*

Answer: c

Title: How many comparisons does selection sort make?

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

26) In Big-Oh notation, selection sort is a(n) \_\_\_\_ algorithm.

a) *O*(*n*2)

b) *O*(1)

c) log *n*

d) *O*(log *n*2)

Answer: a

Title: In Big-Oh notation, selection sort is a(n) \_\_\_\_ algorithm.

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

27) When the size of an array increases by a factor of 100, the time required by selection sort increases by a factor of \_\_\_\_.

a) 2,000

b) 5,000

c) 10,000

d) 12,000

Answer: c

Title: When the size increases by a factor of 100, selection sort time increases by a factor of \_\_\_\_.

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

28) Which notation, big-Oh, theta, or omega describes the growth rate of a function?

I big-Oh

II theta

III omega

a) I

b) II and III

c) I and II

d) I, II, and III

Answer: d

Title: Which notation, big-Oh, theta, or omega, describes the growth rate of a function?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

29) If *f*(*n*) = *O*(*g*(*n*)) and *g*(*n*) = *O*(*f*(*n*)), what else must be true?

I *f*(*n*) = Ω(*g*(*n*))

II *g*(*n*) = Ω(*f*(*n*))

III *f*(*n*) = θ(*g*(*n*))

a) I

b) II

c) I and II

d) I, II, and III

Answer: d

Title: If *f*(*n*) = *O*(*g*(*n*)) and *g*(*n*) = *O*(*f*(*n*)), what else must be true?

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

30) In general, the expression \_\_\_\_ means that *f* grows no faster than *g*.

a) *f*(*n*) = log *g*

b) *f*(*n*) = log *g*2

c) *f*(*n*) = *O*(*g*(*n*))

d) *g*(*n*) = *O*(*f*(*n)*)

Answer: c

Title: The expression \_\_\_\_ means that *f* grows no faster than *g*.

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

31) Find the simplest order of growth of the following expression: (*n*3 + *n* + 3)2.

a) θ(*n*5)

b) θ(2*n*)

c) θ(*n*6)

d) θ(*n*9)

Answer: c

Title: Find the simplest order of growth of the following expression: (n3 + n + 3)2.

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

32) Find the simplest order of growth of the following expression: *n*3 + log(*n*5).

a) θ(*n*3)

b) θ(*n*3+ log(*n*))

c) θ(log(*n*5))

d) θ(*n*+ log(*n*))

Answer: a

Title: Find the simplest order of growth of the following expression: *n*3 + log(*n*5).

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

33) Choose the order of the following growth rates, from slowest to fastest: θ(*n*3), θ(*n*log(*n*)), θ(n3/2), θ(2*n*).

a) θ(*n* log(*n*)), θ(n3/2), θ(*n*3), θ(2*n*)

b) θ(*n*3), θ(*n* log(*n*)), θ(*n*3/2), θ(2*n*)

c) θ(*n*3/2), θ(*n* log(*n*)), θ(*n*3), θ(2*n*)

d) θ(2*n*), θ(*n*3), θ(*n*3/2), θ(*n* log(*n*))

Answer: a

Title: Choose the order of the following growth rates, from slowest to fastest.

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

34) Which function has a faster growth rate: θ(*n*1/2) or θ(log(*n*))?

a) θ(log(*n*))

b) θ(*n*1/2)

c) They are the same.

d) They can’t be compared.

Answer: b

Title: Which function has a faster growth rate: θ(n1/2) or θ(log(n))?

Difficulty: Hard

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.1 Oh, Omega, and Theta

35) Consider the following code snippet:

public static void sort(int[] a)

{

for (int i = 1; i < a.length; i++)

{

int next = a[i];

int j = i;

while (j > 0 && a[j - 1] > next)

{

a[j] = a[j - 1];

j--;

}

a[j] = next;

}

}

What sort algorithm is used in this code?

a) insertion sort

b) selection sort

c) merge sort

d) quicksort

Answer: a

Title: What sort algorithm is used in this code?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.2 Insertion Sort

36) Which sort algorithm starts with an initial sequence of size 1, which is assumed to be sorted, and increases the size of the sorted sequence in the array in each iteration?

a) insertion sort

b) selection sort

c) merge sort

d) quicksort

Answer: a

Title: Which sort algorithm increases the size of the sorted sequence in each iteration?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.2 Insertion Sort

37) What is the worst-case performance of insertion sort?

a) *O*(*n*)

b) *O*(*n*2)

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

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

Answer: b

Title: What is the worst-case performance of insertion sort?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.2 Insertion Sort

38) If the array is already sorted, what is the performance of insertion sort?

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(log *n*)

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

Answer: a

Title: What is the performance of insertion sort on a sorted array?

Difficulty: Medium

Section Reference 1: 14.3 Analyzing the Performance of the Selection Sort Algorithm

Section Reference 2: Special Topic 14.2 Insertion Sort

39) Which sort algorithm starts by cutting the array in half and then recursively sorts each half?

a) insertion sort

b) selection sort

c) merge sort

d) quicksort

Answer: c

Title: Which sort algorithm cuts the array in half and then recursively sorts each half?  
Difficulty: Medium

Section Reference 1: Special Topic 14.4 Merge Sort

40) How many times can an array with 4,096 elements be cut into two equal pieces?

a) 16

b) 12

c) 10

d) 8

Answer: b

Title: How many times can an array with 4,096 elements be cut into two equal pieces?

Difficulty: Easy

Section Reference 1: 14.4 Merge Sort

41) How many times can an array with 729 elements be cut into three equal pieces?

a) 4

b) 5

c) 6

d) 7

Answer: c

Title: How many times can an array with 729 elements be cut into three equal pieces?

Difficulty: Medium

Section Reference 1: 14.4 Merge Sort

42) The merge sort algorithm presented in section 14.4, which sorts an array of integers in ascending order, uses the merge method which is partially shown below. Select the condition that would be needed to complete the method so that the elements are sorted in descending order.

private static void merge(int[] first, int[] second, int[] a)

{

int iFirst = 0;

int iSecond = 0;

int j = 0;

while (iFirst < first.length && iSecond < second.length)

{

if (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

{

a[j] = first[iFirst];

iFirst++;

}

else

{

a[j] = second[iSecond];

iSecond++;

}

j++;

}

// rest of the method follows here

}

a) first[iFirst] < second[iSecond]

b) iFirst < iSecond

c) first[iFirst] > second[iSecond]

d) iFirst > iSecond

Answer: c

Title: Complete merge method to sort array in descending order

Difficulty: Medium

Section Reference 1: 14.4 Merge Sort

43) In the textbook, we determined that the merge method requires a total of 5*n* visits. We found that the number of visits required to sort an array of *n* elements is T(*n*) = T(*n* / 2) + T(*n* / 2) + 5*n*. What does T(*n* / 2) describe?

a) the total number of visits to merge sort an array of *n* elements

b) half the number of visits to merge sort an array of *n* elements

c) the total number of visits to merge sort an array of *n* / 2 elements

d) half the number of visits to merge sort an array of *n* / 2 elements

Answer: c

Title: What does T(*n*/2) describe?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

44) In the textbook, we found that the number of element visits for merge sort totaled   
*n* + 5*n*log2*n*. Let’s consider sorting 1024 elements. How many visits are needed?

a) 1.024

b) 6,144

c) 51,200

d) 52,224

Answer: d

Title: How many visits are needed for a merge sort of 1024 elements?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

45) In the textbook, we found that the number of element visits for merge sort totaled   
*n* + 5*n* log2 *n*. Which of the following is the appropriate big-Oh notation for merge sort?

a) 5*n* log2 *n*

b) *n* + log2 *n*

c) *n* + 5*n*

d) *n* log2 *n*

Answer: d

Title: Which of the following is the appropriate big-Oh notation for merge sort?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

46) Selection sort has *O*(*n*2) complexity. If a computer can sort 1,000 elements in 4 seconds, approximately how many seconds will it take the computer to sort 1,000 times that many, or 1,000,000 elements?

a) 16

b) 1,000

c) 1,000,000

d) 4,000,000

Answer: d

Title: How many seconds will it take selection sort to sort 1,000 times as many elements?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

47) Merge sort has a *O*(*n* log2(*n*)) complexity. If a computer can sort 1,024 elements in an amount of time *x*, approximately how much longer will it take the computer to sort 1,024 times that many, or 1,048,576 elements?

a) 1,024 times longer

b) 2,048 times longer

c) 8,192 times longer

d) 1,048,576 times longer

Answer: c

Title: How much longer will it take a computer to sort these elements?

Difficulty: Hard

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

48) Merge sort is a(n) \_\_\_\_ algorithm.

a) *O*(*n*)

b) *O*(*n* log(*n*))

c) *O*(*n*2)

d) O(log *n*)

Answer: b

Title: Merge sort is a(n) \_\_\_\_ algorithm.

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

49) Assume we are using quicksort to sort an array in ascending order. Into which array location does quicksort’s strategy place a pivot element after partitioning?

a) lowest index in array still available

b) highest index in array still available

c) closer to its correct final location

d) its final correct location

Answer: d

Title: Into which array location does quicksort place the pivot element after partitioning?

Difficulty: Easy

Section Reference 1: Special Topic 14.3 The Quicksort Algorithm

50) Assume we are using quicksort to sort an array in ascending order. What can we conclude about the indexes of two pivot elements placed in consecutive recursive calls?

a) They are randomly located.

b) They are in different halves of the array.

c) They are both in the same half of the array.

d) They are adjacent.

Answer: a

Title: What can we conclude about two pivot elements placed in consecutive recursive calls?

Difficulty: Easy

Section Reference 1: Special Topic 14.3 The Quicksort Algorithm

51) Assume we are using quicksort to sort an array in ascending order. What can we conclude about the elements to the left of the currently placed pivot element?

a) They are all sorted.

b) They are all less than or equal to the pivot element.

c) They are all greater than or equal to the pivot element.

d) None can equal the pivot element.

Answer: b

Title: What can we conclude about the elements to the left of the pivot element in quicksort?

Difficulty: Easy

Section Reference 1: Special Topic 14.3 The Quicksort Algorithm

52) When does quicksort’s worst-case run-time behavior occur?

I when the data is randomly initialized in the array

II when the data is in ascending order

III when the data is in descending order

a) I

b) II

c) III

d) II and III

Answer: d

Title: When does quicksort’s worst-case runtime behavior occur?

Difficulty: Easy

Section Reference 1: Special Topic 14.3 The Quicksort Algorithm

53) Which sort algorithm starts by partitioning the array and selecting a pivot element?

a) merge sort

b) quicksort

c) selection sort

d) insertion sort

Answer: b

Title: Which sort algorithm starts by partitioning the array and selecting a pivot element?

Difficulty: Easy

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

54) A version of which sort algorithm is used in the sort method in the Java Arrays class?

a) merge sort

b) quicksort

c) selection sort

d) insertion sort

Answer: b

Title: A version of which sort algorithm is used in the Arrays.sort method?

Difficulty: Easy

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

55) Which sort algorithm is used in the sort method in the Java Arrays class when the array length is less than 7?

a) merge sort

b) quicksort

c) selection sort

d) insertion sort

Answer: d

Title: Which sort implementation is used in Arrays.sort when length is less than 7?

Difficulty: Easy

Section Reference 1: Special Topic 14.3 The Quicksort Algorithm

56) Which of the sorts in the textbook are based on the strategy of divide and conquer?

I quicksort

II mergesort

III insertion sort

a) I

b) II

c) III

d) I and II

Answer: d

Title: Which of the sorts in the textbook are based on the divide and conquer strategy?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

57) Which of the sorts in the textbook can be characterized by the fact that the best case will have a running time of θ(*n*) if the data is already sorted?

I quicksort

II selection sort

III insertion sort

a) I

b) II

c) III

d) I and III

Answer: c

Title: Which of the sorts’ best case will have a running time of θ(*n*) if the data issorted?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

58) Which of the sorts in the textbook can be characterized by the fact that even in the worst case the running time will be *O*(*n* log(*n*)))?

I quicksort

II selection sort

III merge sort

a) I

b) II

c) III

d) I and III

Answer: c

Title: Which sort’s worst case running time will be *O*(*n* log(*n*)))?

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

59) In the worst case, quicksort is a(n) \_\_\_\_ algorithm.

a) *O*(*n*)

b) *O*(log(*n*))

c) *O*(*n*2)

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

Answer: c

Title: In the worst case, Quicksort is a(n) \_\_\_\_ algorithm.

Difficulty: Medium

Section Reference 1: 14.5 Analyzing the Merge Sort Algorithm

Section Reference 2: Special Topic 14.3 The Quicksort Algorithm

60) In the worst case, a linear search locates a value in an array of length *n* in \_\_\_\_ steps.

a) *O*(*n*2*)*

b) *O*(log *n)*

c) *O*(*n*)

d) *O*(log *n*2)

Answer: c

Title: In the worst case, a linear search locates a value in an array of length *n* in \_\_\_\_ steps.

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

61) The following code is an example of a \_\_\_ search.

public static int search(int[] a, int v)

{

for (int i = 0; i < a.length; i++)

{

if (a[i] == v) { return i; }

}

return -1;

}

a) sorted

b) binary

c) linear

d) random

Answer: c

Title: This code is an example of a \_\_\_\_\_ search.

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

62) Suppose you have a phone number and need to find the address that it corresponds to. If there are 2,000,000 entries, how many do you expect to search in a printed phone directory before finding the address you are looking for?

a) Approximately 50,000 records.

b) Approximately 75,000 records.

c) Approximately 1,000,000 records.

d) Approximately 1,200,000 records.

Answer: c

Title: How many phone book entries would you expect to search?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

63) If an element is present in an array of length *n*, how many element visits, in the worst case, are necessary to find it using a linear search?

a) *n* / 2

b) *n*

c) 2*n*

d) *n*2

Answer: b

Title: If an element is present, how many visits are necessary to find it using a linear search?

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

64) If an element is present in an array of length *n*, how many element visits, on average, are necessary to find it using a linear search?

a) *n* / 2

b) *n*

c) 2*n*

d) *n*2

Answer: a

Title: If an element is present, how many visits, on average, are needed to find it using linear search?

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

65) Another name for linear search is \_\_\_\_ search.

a) sorted

b) sequential

c) random

d) binary

Answer: b

Title: Another name for linear search is \_\_\_\_ search.

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

66) If you implement a recursive linear search, its performance will be \_\_\_\_.

a) *O*(*n*)

b) *O*(*n*2)

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

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

Answer: a

Title: If you implement a recursive linear search, its performance will be \_\_\_\_.

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.1 Linear Search

67) Binary search is an \_\_\_\_ algorithm.

a) *O*(*n*)

b) *O*(*n*2)

c) *O*(log *n*)

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

Answer: c

Title: Binary search is an \_\_\_\_ algorithm

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

68) Given an ordered array with 15 elements, how many elements must be visited in the worst case of binary search?

a) 8

b) 4

c) 3

d) 2

Answer: b

Title: For an ordered array with 15 elements, how many visits in the worst case of binary search?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

69) Given an ordered array with 31 elements, how many elements must be visited in the worst case of binary search?

a) 16

b) 8

c) 5

d) 4

Answer: c

Title: For an ordered array with 31 elements, how many visits in the worst case of binary search?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

70) The analysis for the number of visits in a binary search begins with the equation,   
T(*n*) = T(*n* / 2) + 1. What does the number 1 represent in this equation?

a) One element visit before a recursive call on one half of the array.

b) The total number of recursions required.

c) The fact that the number of elements is odd.

d) The fact that we search either the left half or the right half, but not both.

Answer: a

Title: What does the number 1 represent in this binary search equation?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

71) Suppose we are using binary search on an array with approximately 1,000,000 elements. How many visits should we expect to make in the worst case?

a) 1

b) 16

c) 20

d) 30

Answer: c

Title: For an array with 1,000,000 elements, how many visits in the worst case of binary search?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

72) A binary search is generally \_\_\_\_ a linear search.

a) slower than

b) equal to

c) less efficient than

d) faster than

Answer: d

Title: A binary search is \_\_\_\_ a linear search.

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

73) A search technique where, in each step, you split the size of the search in half is called a\_\_\_\_ search.

a) random

b) binary

c) linear

d) merging

Answer: b

Title: A search that repeatedly splits the size of the search in half is called a \_\_\_\_ search.

Difficulty: Easy

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

74) Can you search the following array using binary search?

int[] A = {6, 5, 4, 2, 0, 1, -1, -17};

a) Yes. Binary search can be applied to any array.

b) No. Binary search can be applied to a sorted array only.

c) Yes, but the algorithm runs slower because the array is in descending order.

d) No, negative numbers are not allowed because they indicate that a value is not present.

Answer: b

Title: Can you search this array using binary search?

Difficulty: Medium

Section Reference 1: 14.6 Searching

Section Reference 2: 14.6.2 Binary Search

75) The partial linear search method below is designed to search an array of String objects. Select the expression that would be needed to complete the method.

public static int search(String[] a, String item)

{

for (int i = 0; i < a.length; i++)

{

if ( \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ )

{

return i;

}

return -1;

}

}

a) a[i] == item

b) a[i].compareTo(item)

c) a[i].equals(item)

d) a[i].indexOf(item)

Answer: c

Title: Complete linear search method to search an array of String objects

Difficulty: Medium

Section Reference 1: 14.6 Searching

76) The partial binary search method below is designed to search an array of String objects sorted in ascending order. Select the expression that would be needed to complete the method.

public static int search(String[] a, int low, int high, String item)

{

if (low <= high)

{

int mid = (low + high) / 2;

int result = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

if (result == 0)

{

return mid;

}

else if (result < 0)

{

return search(a, mid + 1, high, item);

}

else

{

return search(a, low, mid - 1, item);

}

}

return -1;

}

a) a[low].compareTo(item)

b) item.equals(a[mid])

c) item.compareTo(a[mid])

d) a[mid].compareTo(item)

Answer: d

Title: Complete binary search method to search a sorted array of String objects

Difficulty: Medium

Section Reference 1: 14.6 Searching

77) A portion of your program implements a loop in which each step contains a fixed number of actions. What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(*n* log (*n*)).

Answer: a

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

78) Which of the following statements about running times of algorithms is correct?

a) An algorithm that is *O*(1) means that only one comparison takes place.

b) When determining the running time, constants are not taken into consideration.

c) When determining the running time, lower-order terms must be taken into consideration.

d) An algorithm that is *O*(*n*) means that the number of comparisons does not grow as the size of the array increases.

Answer: b

Title: Which of the following statements about running times of algorithms is correct?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

79) A portion of your program includes the loop shown in the code snippet below to examine the elements of an array arr:

int count = 0;

int targetVal = 70;

for (int i = 0; i < arr.length; i++)

{

if (arr[i] >= targetVal)

{

count++;

}

}

What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(*n* log (*n*)).

Answer: a

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

80) The method findLargest examines the elements of an array arr which contains non-negative values

public static int findLargest(int[] arr)

{

int curLargest = -1;

for (int i = 0; i < arr.length; i++)

{

if (arr[i] >= curLargest)

{

curLargest = arr[i];

}

}

return curLargest;

}

What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(*n* log (*n*)).

Answer: a

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

81) The method checkArray examines an array arr:

public static boolean checkArray(int[] arr)

{

if (arr[0] >= arr[arr.length -1])

{

return true;

}

return false;

}

What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(1).

Answer: d

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

82) An algorithm that tests whether the first array element is equal to any of the other array elements would be an \_\_\_\_ algorithm.

a) *O*(1)

b) *O*(*n*)

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

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

Answer: b

Title: Testing whether the first element is equal to any otherelement would be an \_\_\_\_ algorithm.

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.1 Linear Time

83) A portion of your program includes the loops shown in the code snippet below to examine the elements of two arrays, arr1 and arr2, both of length *n*:

int matches = 0;

for (int i = 0; i < arr1.length; i++)

{

for (int j = 0; j < arr2.length; j++)

{

if (arr1[i].equals(arr2[j]))

{

matches++;

}

}

}

What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(*n* log (*n*)).

Answer: b

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.2 Quadratic Time

84) A portion of your program includes the method shown in the code snippet below to examine the elements of an array arr:

private int findElement(int[] arr, int newVal)

{

int pos = Arrays.binarySearch(arr, newVal);

return pos;

}

What can you conclude about the running time of this section of code?

a) Its running time will be *O*(*n*).

b) Its running time will be *O*(*n*2).

c) Its running time will be *O*(log (*n*)).

d) Its running time will be *O*(*n* log (*n*)).

Answer: c

Title: What can you conclude about the running time of this section of code?

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.4 Logarithmic Time

85) An algorithm that cuts the work in half in each step is an \_\_\_\_ algorithm.

a) *O*(*n*)

b) *O*(*n*2)

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

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

Answer: c

Title: An algorithm that cuts the work in half in each step is an \_\_\_\_ algorithm.

Difficulty: Easy

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

Section Reference 2: 14.7.4 Logarithmic Time

86) The code segment below prints some of the elements in an array with size n. Select an expression to complete the code segment so that the resulting algorithm has *O(log n)* running time.

for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

System.out.println(array[j]);

}

a) (int j = 0; j < array.length; j = j + 2)

b) (int j = 1; j < array.length; j = j \* 2)

c) (int j = 0; j < array.length / 2; j++)

d) (int j = 0; j < array.length; j++)

Answer: b

Title: Select an expression to complete a logarithmic time algorithm

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

87) The code segment below displays a pattern of asterisks. Select an expression to complete the code segment so that the resulting algorithm has *O(n)* running time.

for (int k = 0; k < n; k++)

{

for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

System.out.print("\*");

}

System.out.println();

}

a) (int j = n; j > 0; j = j / 2)

b) (int j = n; j > 0; j--)

c) (int j = 1; j < k; j++)

d) (int j = 1; j <= 10; j++)

Answer: d

Title: Select an expression to complete a linear time algorithm

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

88) The code segment below displays a table of numbers. Select an expression to complete the code segment, so that the resulting algorithm has *O(n2)* running time.

for (int k = 1; k <= n; k++)

{

for \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

{

System.out.print(j + " ");

}

System.out.println();

}

a) (int j = n; j > 0; j = j / 2)

b) (int j = 1; j < k; j = j \* 2)

c) (int j = 0; j < k; j++)

d) (int j = 1; j <= 200; j++)

Answer: c

Title: Select an expression to complete a quadratic time algorithm

Difficulty: Medium

Section Reference 1: 14.7 Problem Solving: Estimating the Running Time of an Algorithm

89) Assume that names is an array of String objects that has been initialized with a large number of elements. Select the statement that would sort the elements in names in ascending alphabetic order.

a) Arrays.sort(names, 0, names.length - 1);

b) Arrays.sort(names);

c) Collections.sort(names, 0, names.length - 1);

d) Collections.sort(names);

Answer: b

Title: Select a statement to sort an array of String objects using Java library method

Difficulty: Easy

Section Reference 1: 14.8 Sorting and Searching in the Java Library

90) Assume that bands is an ArrayList of String objects which contains a number of elements in ascending order. Select a statement to complete the code segment below, which invokes the Java library binarySearch method to search for the string "Beatles". If the list does not already contain the string, it should be inserted in an appropriate location so that the list remains sorted.

int index = Collections.binarySearch(bands, "Beatles");

if (index < 0)

{

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

}

a) bands.add(-1 \* index + 1, "Beatles");

b) bands.add(index + 1, "Beatles");

c) bands.add(-1 \* index, "Beatles");

d) bands.add(-1 - index, "Beatles");

Answer: d

Title: Select a statement to insert a string into a sorted ArrayList

Difficulty: Hard

Section Reference 1: 14.8 Sorting and Searching in the Java Library

91) The sort method of the Arrays class sorts arrays containing objects of classes that implement the \_\_\_\_ interface.

a) Sortable

b) Ordered

c) Array

d) Comparable

Answer: d

Title: Arrays.sort sorts objects of classes that implement the \_\_\_\_ interface.

Difficulty: Easy

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.1 Sorting

92) The \_\_\_\_ class contains a sort method that can sort array lists.

a) Sorting

b) Arrays

c) Collections

d) Linear

Answer: c

Title: The \_\_\_\_ class has a sort method that can sort array lists.

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.1 Sorting

93) The binarySearch method of the Collections class returns a value in the form of –k - 1 when the target item you are searching for was not found in the array. What does k represent?

a) It is the position of an existing item, and indicates that your target item should come after this existing item.

b) It is the position of an existing item, and indicates that your target item should come before this existing item.

c) It represents the number of times the array was accessed by the binarySearch method, and can be used for analyzing performance.

d) Nothing – it is an arbitrary value.

Answer: b

Title: What does the value k represent when returned from Collections.binarySearch?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

94) Given the following code snippet for searching an array:

int[] arr = {3, 8, 12, 14, 17};

int newVal = 15;

int pos = Arrays.binarySearch(arr, newVal);

What value will pos have when this code is executed?

a) 5

b) -5

c) 6

d) -6

Answer: b

Title: What value will pos have when this code is executed?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

95) Given the following code snippet for searching an array:

int[] arr = {23, 25, 29, 34, 42};

int newVal = 15;

int pos = Arrays.binarySearch(arr, newVal);

What value will pos have when this code is executed?

a) 0

b) 1

c) -1

d) -2

Answer: c

Title: What value will pos have when this code is executed?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

96) If a call to the Arrays static method binarySearch returns a value of -10, what can be concluded?

I the element is not in the array

II the element is at index 10

III the element can be inserted at index 9

a) I

b) II

c) III

d) I and III

Answer: d

Title: If a call to the Arrays.binarySearch returns a value of -10, what can be concluded?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

97) If a call to the Arrays static method binarySearch returns a value of 7, what can be concluded?

I the element is not in the array

II the element is at index 7

III the element occurs 7 times in the array

a) I

b) II

c) III

d) II and III

Answer: b

Title: If a call to the Arrays.binarySearch returns a value of 7, what can be concluded?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

98) If the Arrays static method binarySearch is called on an array of 10 elements and returns a value of 10, what can be concluded?

I the element is not found

II that value cannot be returned from method binarySearch

III if added, the element would be the largest element in the array

a) I

b) II

c) III

d) I and III

Answer: b

Title: If Arrays.binarySearch called on an array of 10 elements returns a value of 10, what can be concluded?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.2 Binary Search

99) If you want to use the Comparable interface, you must implement a single method called \_\_\_\_.

a) comparable

b) compareTo

c) comparator

d) compare

Answer: b

Title: To use the Comparable interface, you must implement a method called \_\_\_\_.

Difficulty: Easy

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.3 Comparing Objects

100) Which of the following classes implement the Comparable interface?

I Date

II Collections

III String

a) I

b) I and II

c) I and III

d) II and III

Answer: c

Title: Which of the following classes implement the Comparable interface?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.3 Comparing Objects

101) Suppose the call obj1.compareTo(obj2) returns 0. What can definitely be concluded from the return value?

I obj1 and obj2 are the same object

II obj1 and obj2 objects cannot be compared

III the properties of obj1 and obj2 that are being compared have identical values

a) I

b) II

c) I and III

d) III

Answer: d

Title: Suppose the call obj1.compareTo(obj2) returns 0. What can definitely be concluded from the return value?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.3 Comparing Objects

102) Which of the following arrays can be used in a call to the Arrays.sort method?

I Any array with primitive numeric data, such as int, double, …

II Arrays of String or numeric wrapper classes like, Integer, Double, …

III Any class that implements the Comparable interface

a) I

b) II

c) I and II

d) I, II and III

Answer: d

Title: Which of the following arrays can be used in a call to the Arrays.sort method?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: 14.8.3 Comparing Objects

103) Suppose objects a and b are from a user-defined class that implements the Comparable interface. Which condition tests the compareTo method’s return value to determine that a will precede b when the sort method is called?

a) a.compareTo(b) == -1

b) a.compareTo(b) == 0

c) a.compareTo(b) < 0

d) a.compareTo(b) > 0

Answer: c

Title: Which condition tests compareTo’s return value to determine that a will precede b?

Difficulty: Easy

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Common Error 14.1 The compareTo Method Can Return Any Integer, Not Just -1, 0, and 1

104) Suppose objects a and b are from a user-defined class that implements the Comparable interface. What must be true about the return value of a.compareTo(b) for the compareTo method that this class implements?

a) It must return 1 if a comes before b, 0 if they are the same, and -1 if a comes after b.

b) It must return -1 if a comes before b, 0 if they are the same, and 1 if a comes after b.

c) It must return a positive value if a comes before b, 0 if they are the same, and a negative value if a comes after b.

d) It must return a negative value if a comes before b, 0 if they are the same, and a positive value if a comes after b.

Answer: d

Title: What must be true about the return value of the compareTo method?

Difficulty: Hard

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Common Error 14.1 The compareTo Method Can Return Any Integer, Not Just -1, 0, and 1

105) Suppose you wish to implement the Comparable interface to allow your Vehicle class to compare Auto objects only. Which of the following is the correct way to do this?

a) public class Auto implements Comparable<Vehicle>

b) public class Vehicle implements Comparable

c) public class Vehicle implements Comparable<Auto>

d) public class Vehicle implements Comparable(Auto)

Answer: c

Title: How to specify that one class will compare objects of another class

Difficulty: Hard

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.4 The Parameterized Comparable Interface

106) Suppose a developer gets class XYZ files and documentation from a subcontractor. This class does not implement the Comparable interface. What must be true in order for the developer to sort an array of XYZ objects without modifying the xyz class?

a) The XYZ class must implement the Sortable interface.

b) XYZ objects must be randomly distributed.

c) XYZ objects must be ordered.

d) The developer must supply a comparator object belonging to a class that implements the Comparator<XYZ> interface.

Answer: d

Title: How does the developer sort an array of XYZ objects that do not implement Comparable?

Difficulty: Hard

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.5 The Comparator Interface

107) Suppose you wish to sort an array list of objects, but the object class does not implement the Comparable interface. Because you are not allowed to modify this class, you decide to provide a comparator object that implements the Comparator interface. Which method must you implement from this interface to achieve your objective?

a) the sort method.

b) the compare method.

c) the compareTo method.

d) the compareObject method.

Answer: b

Title: Which method of the Comparator interface must you implement?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.5 The Comparator Interface

108) Complete the following code for an interface that classes who wish to compare Auto objects can implement.

public interface Comparator<Auto>

{

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

}

a) boolean compare(Auto a, Auto b);

b) single compareTo(Auto a, Auto b);

c) int compare(Auto a, Auto b);

d) single compare(Auto a, Auto b);

Answer: c

Title: Complete the code for a comparator interface.

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.5 The Comparator Interface

109) When your class implements a comparator object, you must implement the compare method. What must be true about the return value from this method when comparing two objects, a and b with a call to a.compare(b)?

a) It must return 1 if a comes before b, 0 if they are the same, and -1 if a comes after b.

b) It must return -1 if a comes before b, 0 if they are the same, and 1 if a comes after b.

c) It must return a positive value if a comes before b, 0 if they are the same, and a negative value if a comes after b.

d) It must return a negative value if a comes before b, 0 if they are the same, and a positive value if a comes after b.

Answer: d

Title: What must be true about the return value from the compare method of a comparator object?

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.5 The Comparator Interface

110) Complete the following code that is intended to provide a comparator interface that will be used to create an object that implements the Comparator interface so that object can compare Auto objects.

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

{

int compare(Auto a, Auto b);

}

a) public class Comparator<Auto>

b) public class Comparator<Auto> implements Comparator

c) public interface Auto<Comparator>

d) public interface Comparator<Auto>

Answer: d

Title: Complete the code for a comparator interface.

Difficulty: Medium

Section Reference 1: 14.8 Sorting and Searching in the Java Library

Section Reference 2: Special Topic 14.5 The Comparator Interface