Exam 1

Due Feb 18, 2021 at 10:45am **Points** 100 **Questions** 34
Available until Feb 18, 2021 at 10:45am **Time Limit** 75 Minutes

Instructions

This exam consists entirely of multiple-choice questions and has a 75-minute time limit. For each question, choose the **best** response of those given.

(https(

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

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	31 minutes	97 out of 100

Score for this quiz: **97** out of 100
Submitted Feb 18, 2021 at 10:01am
This attempt took 31 minutes.

Question 1 — 3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    int i = 100;
    while (i > 1) {
        i = i - 2;
        for (int j = 0; j < 100; j++) {
            for (int k = 0; k < 100; k++) {
                count++;
            }
        }
    }
    return count;
}
```

A. $O(N^3)$

B. $O(N^2)$

C. $O(N)$

D. $O(1)$

- ○ A
- ○ B
- ○ C
- ◉ D

Correct!

Question 2

3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    int i = N;
    while (i > 1) {
        i = i - 2;
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < N; k++) {
                count++;
            }
        }
    }
    return count;
}
```

A. $O(N^3)$

B. $O(N^2)$

C. $O(N)$

D. $O(1)$

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D

Question 3

3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    int i = N;
    while (i > 1) {
        count++;
        i = i / 2;
    }
    i = N;
    while (i > 1) {
        count++;
        i = i - 2;
    }
    return count;
}
```

A. $O(\log N)$

B. $O(N)$

C. $O(N \log N)$

D. $O(\log \log N)$

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 4

3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    int i = N;
    for (int j = 1; j < 10; j++) {
        count++;
    }
    while (i > 1) {
        count++;
        i = i / 2;
    }
    return count;
}
```

A. $O(\log N)$

B. $O(N)$

C. $O(N \log N)$

D. $O(N^2)$

Correct!

- (selected) A
- B
- C
- D

Question 5

3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    for (int i = 1; i < N; i = i + 2) {
        int j = 1;
        while (j < N) {
            count++;
            j = j * 2;
        }
    }
    return count;
}
```

A. $O(\log \log N)$

B. $O(\log N^2)$

C. $O(N \log N)$

D. $O(N^2)$

- ○ A
- ○ B
- ◉ C

Correct!

- ○ D

Question 6

3 / 3 pts

Which big-O expression best characterizes the time complexity of the following code?

```
public static int foo(int N) {
    int count = 0;
    int j = 0;
    while (j < N) {
        for (int i = 1; i < N; i = i + 1) {
            count++;
        }
        j = j + 2;
    }
    return count;
}
```

A. $O(\log \log N)$

B. $O(\log N^2)$

C. $O(N \log N)$

D. $O(N^2)$

- A
- B
- C
- Correct! D

Question 7

3 / 3 pts

Suppose you attempted to empirically discover the big-O running time of a program, and you were able to generate the following timing data.

N	Time	Ratio
128	0.513	—
256	3.223	6.283
512	23.444	7.274
1024	186.849	7.970
2048	1493.114	7.991

In the table above, the N column records the size of the input for each run, the Time column records the elapsed time in seconds for each run, and Ratio is the elapsed time for the current run divided by the elapsed time for the previous run (i.e., $Time_{\text{i}}/Time_{\text{i-1}}$).

Based on the timing data presented in this table, what is the most reasonable conclusion regarding the underlying big-O time complexity of the program being timed?

A. $O(8)$

B. $O(\log_3 N)$

C. $O(N^3)$

D. $O(N^8)$

- A
- B
- C

Correct!

- D

Question 8

3 / 3 pts

Suppose you have implemented an algorithm in a method named `foo`, which takes an array of N floating point numbers as data. Suppose also that this algorithm has $O(N^4)$ best, average, and worst case time complexity, and that a timing analysis of `foo` showed that approximately 1 second was required to process an array of size $N = 125$.

What is the largest array (N) that `foo` could process in less than one hour?

A. 250

B. 500

C. 1000

D. 2000

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 9

3 / 3 pts

Which big-oh expression best characterizes the worst case time complexity of the following implementation of `kmin`?

```
/**
 * Returns the kth smallest element in an array.
 */
public static int kmin(int[] a, int k) {
   for (int i = 0; i < k; i++) {
      int min = i;
      for (int j = i + 1; j < a.length; j++) {
         if (a[j] < a[min]) {
            min = j;
         }
      }
      swap(a, i, min);
   }
   return a[k];
}
```

A. $O(N^2)$

B. $O(N \log N)$

C. $O(N)$

D. $O(1)$

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D

Question 10

0 / 3 pts

What is the maximum number of times the comparison `a[middle] == target)` would be executed if `a.length` is N and `target` is some value not in `a`?

```
public int binarySearch(int[] a, int target) {
    int left = 0;
    int right = a.length - 1;
    while (left <= right) {
        int middle = left + (right - left) / 2;
        if (a[middle] == target) {
            return middle;
        }
        if (a[middle] > target) {
            right = middle - 1;
        } else {
            left = middle + 1;
        }
    }
    return -1;
}
```

A. $\log_2 N$

B. $\lfloor \log_2 N \rfloor + 1$

C. $N/2$

D. N

E. There isn't enough information given to answer the question.

ou Answered

- (•) A

orrect Answer

- ( ) B
- ( ) C
- ( ) D
- ( ) E

Question 11

3 / 3 pts

Which big-oh expression best characterizes the worst case time complexity of the following implementation of `kmin`?

```
/**
 * Returns the kth smallest element in an
 *  array of unique values.
 */
public Comparable kmin(Comparable[] a, int k) {
   java.util.Arrays.sort(a);
   return a[k - 1];
}
```

A. $O(1)$

B. $O(N)$

C. $O(N \log N)$

D. $O(N^2)$

- ○ A
- ○ B
- Correct! ◉ C
- ○ D

Question 12

3 / 3 pts

The `max` method below is intended to return the largest value in the parameter `coll`. Which test case parameter would expose the logic error in this method?

```
public static <T extends Comparable<T>> T max(Collection<T> coll)
   Iterator<T> itr = coll.iterator();
   T max = itr.next();
   while (itr.hasNext()) {
      if (max.compareTo(itr.next()) < 0) {
         max = itr.next();
      }
   }
   return max;
}
```

A. `coll = [1, 3, 5, 7, 9]`

B. `coll = [9, 7, 5, 3, 1]`

C. `coll = [3, 7, 9, 1, 5]`

D. `coll = [1, 9, 3, 7, 5]`

- ○ A
- ○ B
- ○ C
- ◉ D

Correct!

Question 13

3 / 3 pts

The `min` method below is intended to return the smallest of its three `int` parameters. Which call below would expose the logic error in this method?

```
public static int min(int a, int b, int c) {
   if ((a < b) && (a < c)) {
      return a;
   }
   if ((b < a) && (b < c)) {
      return b;
   }
   return c;
}
```

A. `min(0, 0, 0)`

B. `min(1, 0, -1)`

C. `min(4, 2, 2)`

D. `min(2, 2, 4)`

- ○ A
- ○ B
- ○ C
- ◉ D

Correct!

Question 14

3 / 3 pts

The `max` method below is intended to return the largest value in the parameter `a`. Which test case parameter would expose the logic error in this method?

```
public static int max(int[] a) {
    int max = -1;
    for (int value : a) {
        if (value > max) {
            max = value;
        }
    }
    return max;
}
```

A. `coll = [0, 0, 0, 0, 0]`

B. `coll = [-3, -1, 0, 1, 3]`

C. `coll = [-3, -5, -7, -9, -11]`

D. `coll = [1, 3, 5, 7, 9]`

- ○ A
- ○ B
- Correct! ◉ C
- ○ D

Question 15

3 / 3 pts

The `search` method below is intended to return the index of `target` in `a` or -1 if `a` does not contain `target`. If `a = [1, 3, 5, 7, 9]` which value of `target` below would expose the logic error in this method?

```
public static int search(int[] a, int target) {
    int i = 0;
    while ((a[i] != target) && (i < a.length)) {
        i++;
    }
    if (a[i] == target) {
        return i;
    }
    else {
        return -1;
    }
}
```

A. 1

B. 5

C. 8

D. 9

- ○ A
- ○ B
- ◉ C

Correct!

- ○ D

Question 16

3 / 3 pts

The `kmin` method below is intended to return the k^{th} smallest value in the parameter `list`. Which test case parameters would expose the logic error in this method?

```
public static Integer kmin(List<Integer> list, int k) {
   Collections.<Integer>sort(list);
   return list.get(k - 1);
}
```

A. `list = [7, 3, 5, 1, 7], k = 3`

B. `list = [3, 1, 3, 5, 7], k = 3`

C. `list = [1, 3, 5, 7, 9], k = 3`

D. `list = [9, 7, 5, 3, 1], k = 3`

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 17

3 / 3 pts

What is the result of executing the following statements?

```
String s = "War Eagle!";
boolean result = s.equals(null);
System.out.println(result);
```

A. `false` is printed

B. `true` is printed

C. `War Eagle!` is printed

D. A `NullPointerException` is thrown

E. An `IllegalArgumentException` is thrown

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D
- ( ) E

Question 18

3 / 3 pts

Which expression would be best to perform the intended comparison in the method `search` below?

```
public boolean search(Object[] a, Object key) {
    for (Object value : a) {
        if (_____________________) {
            return true;
        }
    }
    return false;
}
```

A. `value == key`

B. `value.equals(key)`

C. `value.compareTo(key) == 0`

D. `value.compareTo(key) == -1`

E. `value.compareTo(key) == 1`

- ( ) A

Correct!

- (●) B
- ( ) C
- ( ) D

E

Question 19

3 / 3 pts

What should we expect `cmp` to contain when the following code executes?

```
String s1 = "A";
String s2 = "F";

int cmp = s1.compareTo(s2);
```

A. 0

B. -1

C. 1

D. some negative integer

E. some positive integer

A

B

C

Correct!

D

E

Question 20

3 / 3 pts

The `min` method below is intended to return the minimum value in the array `a`. What would be the best comparison to fill in the missing `if` condition?

```
public static Object min(Object[] a) {
    Object min = a[0];
    for (Object value : a) {
        if (_____________) {
            min = value;
        }
    }
    return min;
}
```

A. value $<$ min

B. min $<$ value

C. value.compareTo(min) < 0

D. min.compareTo(value) < 0

E. None of the above

- ○ A
- ○ B
- ○ C
- ○ D
- ◉ E

Correct!

Question 21

3 / 3 pts

What would happen if you tried to compile and run the following code?

```
public class MyClass {

    public static void main(String[] args) {
        Comparable[] a = {"hi", new Double(1.14), new Integer(3)};
        Comparable key = new Float(3.14);
        System.out.println(search(a, key));
    }

    public static boolean search(Comparable[] a, Comparable key) {
        for (Comparable value : a) {
            if (value.compareTo(key) == 0) {
                return true;
            }
        }
        return false;
    }

}
```

A. A compile-time error is generated.

B. Compilation succeeds but a runtime error is generated.

C. Compilation succeeds and the program runs without error.

D. Compile-time warnings are issued, but the program runs without error.

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 22

3 / 3 pts

Which method supports type-safety and would compile with no warnings or errors?

A.
```
public <T extends Comparable<T>> boolean search(Collection<T> coll, T target) {
  for (T element : coll) {
    if (element.compareTo(target) == 0) {
      return true;
    }
  }
  return false;
}
```

B.
```
public <T extends Comparable> boolean search(Collection<T> coll, T target) {
  for (T element : coll) {
    if (element.compareTo(target) == 0) {
      return true;
    }
  }
  return false;
}
```

C.
```
public boolean search(Collection<Comparable> coll, Comparable target) {
  for (Comparable element : coll) {
    if (element.compareTo(target) == 0) {
      return true;
    }
  }
  return false;
}
```

D.
```
public <T extends Comparable> boolean search(Collection<Comparable> coll, T target) {
  for (Comparable element : coll) {
    if (element.compareTo(target) == 0) {
      return true;
    }
  }
  return false;
}
```

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D

Question 23

3 / 3 pts

The `min` method below is intended to return the minimum value in the array `a`. What would be the best comparison to fill in the missing `if` condition?

```
public static <T> T min(T[] a, Comparator<T> comp) {
    T min = a[0];
    for (T value : a) {
        if (__________) {
            min = value;
        }
    }
    return min;
}
```

A. min.compareTo(value) < 0

B. value.compareTo(min) < 0

C. comp.compare(min, value) < 0

D. comp.compare(value, min) < 0

E. None of the above

- ○ A
- ○ B
- ○ C
- ◉ D — Correct!
- ○ E

Question 24

3 / 3 pts

Which of the following algorithms was **not** an *in place* sort?

A. selection sort

B. insertion sort

C. merge sort

D. quicksort

A

B

Correct!

C

D

Question 25

3 / 3 pts

Suppose you have an array full of instances of a class named `Person`. Each person has a name and a date of birth. The following is an example of such an array.

```
[Cin 05-08-94, Rob 02-22-93, Abb 08-17-98, Don 02-22-93, Fay
07-11-97, Amy 03-14-96]
```

To arrange the elements in this array in ascending date of birth order, where multiple persons with the same date of birth are listed in ascending order of name, which is the correct sequence of sorts to perform? (An example of the desired order is shown below.)

```
[Don 02-22-93, Rob 02-22-93, Amy 03-14-96, Cin 05-08-94, Fay
07-11-97, Abb 08-17-98]
```

A. Use merge sort to sort the array in ascending order of date of birth, then use merge sort to sort the array in ascending order of name.

B. Use merge sort to sort the array in ascending order of name, then use merge sort to sort the array in ascending order of date of birth.

C. Use quicksort to sort the array in ascending order of date of birth, then use quicksort to sort the array in ascending order of name.

D. Use merge sort to sort the array in ascending order of name, then use quicksort to sort the array in ascending order of date of birth.

A

Correct!

B

C

D

Question 26

3 / 3 pts

Given the array a = [66, 67, 20, 86, 55, 74, 11, 91, 43, 47] which sorting algorithm would perform the following sequence of array modifications?

```
[66, 67, 20, 86, 55, 74, 11, 91, 43, 47]
[20, 66, 67, 86, 55, 74, 11, 91, 43, 47]
[20, 66, 67, 86, 55, 74, 11, 91, 43, 47]
[20, 55, 66, 67, 86, 74, 11, 91, 43, 47]
[20, 55, 66, 67, 74, 86, 11, 91, 43, 47]
[11, 20, 55, 66, 67, 74, 86, 91, 43, 47]
[11, 20, 55, 66, 67, 74, 86, 91, 43, 47]
[11, 20, 43, 55, 66, 67, 74, 86, 91, 47]
[11, 20, 43, 47, 55, 66, 67, 74, 86, 91]
```

A. selection sort

B. insertion sort

C. merge sort

D. quicksort

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 27

3 / 3 pts

Given the array a = [66, 67, 20, 86, 55, 74, 11, 91, 43, 47] which sorting algorithm would perform the following sequence of array modifications?

```
[66, 67, 20, 86, 55, 74, 11, 91, 43, 47]
[20, 66, 67, 86, 55, 74, 11, 91, 43, 47]
[20, 66, 67, 55, 86, 74, 11, 91, 43, 47]
[20, 55, 66, 67, 86, 74, 11, 91, 43, 47]
[20, 55, 66, 67, 86, 11, 74, 91, 43, 47]
[20, 55, 66, 67, 86, 11, 74, 91, 43, 47]
[20, 55, 66, 67, 86, 11, 74, 91, 43, 47]
[20, 55, 66, 67, 86, 11, 43, 47, 74, 91]
[11, 20, 43, 47, 55, 66, 67, 74, 86, 91]
```

A. selection sort

B. insertion sort

C. merge sort

D. quicksort

- ○ A
- ○ B
- ◉ C

Correct!

- ○ D

Question 28

3 / 3 pts

Which of the arrays below would be the final result of *partitioning* the following portion of an array using 59 as the pivot in the quicksort partition implementation presented in lecture? Only the partitioning operation is happening.

`[97, 20, 84, 24, 25, 59, 93, 13, 94]`

A. `[20, 93, 13, 97, 59, 24, 25, 94, 84]`

B. `[94, 93, 97, 84, 59, 20, 24, 25, 13]`

C. `[20, 24, 25, 13, 59, 94, 93, 97, 84]`

D. `[20, 24, 84, 97, 59, 13, 25, 93, 94]`

- ○ A
- ○ B
- ◉ C — Correct!
- ○ D

Question 29

3 / 3 pts

Given the array a = [66, 67, 20, 86, 55, 74, 11, 91, 43, 47] which sorting algorithm would perform the following sequence of array modifications?

```
[11, 67, 20, 86, 55, 74, 66, 91, 43, 47]
[11, 20, 67, 86, 55, 74, 66, 91, 43, 47]
[11, 20, 43, 86, 55, 74, 66, 91, 67, 47]
[11, 20, 43, 47, 55, 74, 66, 91, 67, 86]
[11, 20, 43, 47, 55, 74, 66, 91, 67, 86]
[11, 20, 43, 47, 55, 66, 74, 91, 67, 86]
[11, 20, 43, 47, 55, 66, 67, 91, 74, 86]
[11, 20, 43, 47, 55, 66, 67, 74, 91, 86]
[11, 20, 43, 47, 55, 66, 67, 74, 86, 91]
[11, 20, 43, 47, 55, 66, 67, 74, 86, 91]
```

A. selection sort

B. insertion sort

C. merge sort

D. quicksort

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D

Question 30

3 / 3 pts

Suppose you have collected data from several timing experiments and you have characterized a sorting method's time complexity profile as follows.

- Data in ascending order: $O(N)$
- Data in descending order: $O(N^2)$
- Data in random order: $O(N^2)$

Which sorting algorithm is implemented in this method?

A. selection sort

B. insertion sort

C. merge sort

D. quicksort (without randomization)

- ○ A
- Correct! ◉ B
- ○ C
- ○ D

Question 31

3 / 3 pts

Suppose you have collected data from several timing experiments and you have characterized a sorting method's time complexity profile as follows.

- Data in ascending order: $O(N \log N)$
- Data in descending order: $O(N \log N)$
- Data in random order: $O(N \log N)$

Which sorting algorithm is implemented in this method?

A. selection sort

B. insertion sort

C. merge sort

D. quicksort (without randomization)

- ○ A
- ○ B
- ◉ C

Correct!

- ○ D

Question 32

3 / 3 pts

Suppose you have collected data from several timing experiments and you have characterized a sorting method's time complexity profile as follows.

- Data in ascending order: $O(N^2)$
- Data in descending order: $O(N^2)$
- Data in random order: $O(N \log N)$

Which sorting algorithm is implemented in this method?

A. selection sort

B. insertion sort

C. merge sort

D. quicksort (without randomization)

- ○ A
- ○ B
- ○ C
- ◉ D

Correct!

Question 33

3 / 3 pts

Suppose you have collected data from several timing experiments and you have characterized a sorting method's time complexity profile as follows.

- Data in ascending order: $O(N^2)$
- Data in descending order: $O(N^2)$
- Data in random order: $O(N^2)$

Which sorting algorithm is implemented in this method?

A. selection sort

B. insertion sort

C. merge sort

D. quicksort (without randomization)

Correct!

- (●) A
- ( ) B
- ( ) C
- ( ) D

Question 34

1 / 1 pts

Click the only answer below for an automatic 1 point.

Correct!

- (●) A

Quiz Score: **97** out of 100