Exam 2

● Graded

Student

Andrew Onozuka

Total Points

44.5 / 49 pts

Question 1

Excel with Integrity Pledge

0 / 0 pts

✓ **+ 0 pts** Correct

+ 0 pts Incorrect

Question 2

Instructions

0 / 0 pts

✓ **+ 0 pts** Correct

+ 0 pts Incorrect

Question 3

Partition **12** / 12 pts

3.1 A **2** / 2 pts

✓ **+ 2 pts** Correct

+ 0 pts Incorrect

3.2 B **2** / 2 pts

✓ **+ 2 pts** Correct (Also includes None of the Above due to typo)

+ 2 pts Correct

+ 0 pts Incorrect

3.3 C **2** / 2 pts

+ 0 pts Incorrect

✓ **+ 2 pts** Correct (Also includes None of the Above due to typo)

+ 2 pts Correct

3.4 D **2** / 2 pts

✓ **+ 2 pts** Correct (also includes None of the above due to typo)

+ 2 pts Correct

+ 0 pts Incorrect

3.5 E **2** / 2 pts

✓ **+ 2 pts** Correct

+ 0 pts Incorrect

3.6 F **2** / 2 pts

✓ **+ 2 pts** Correct

+ 0 pts Incorrect

Question 4

Run-time **12** / 12 pts

4.1 A **1** / 1 pt

✓ **+ 1 pt** Correct: n^4

+ 0 pts Incorrect

+ 1 pt Correct

4.2 B **0** / 0 pts

+ 0 pts Correct: n^4

✓ **+ 0 pts** Incorrect

+ 0 pts Correct

4.3 C **0** / 0 pts

+ 0 pts Correct: n^n

+ 0 pts Correct

✓ **+ 0 pts** Incorrect

4.4 D **1** / 1 pt

+ 1 pt Correct: n^2*log(n)

✓ **+ 1 pt** Incorrect - point given due to typo (missing base case)

+ 1 pt Correct

+ 0 pts Incorrect

4.5 E **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.6 F **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.7 G **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.8 H 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.9 I 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.10 J 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.11 K 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.12 L 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.13 M 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

4.14 N 1 / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 5

HashTable **8.5** / 9 pts

5.1 **HasherA - Output** **3** / 3 pts

✓ **− 0 pts** Correct: 1 / 2 / null / 3 / null / 4

− 1 pt One incorrect

− 2 pts Two incorrect

− 3 pts Three or more incorrect

− 1 pt Extra incorrect information in output

5.2 **HasherB - Output** **3** / 3 pts

✓ **− 0 pts** Correct: 1 / 2 / 100 / 2 / 300 / 3

− 1 pt One incorrect

− 2 pts Two incorrect

− 3 pts Three or more incorrect

− 1 pt Extra incorrect information in output

5.3 **HasherA - Buckets** **1** / 1 pt

✓ **+ 1 pt** Correct: 3

+ 0 pts Incorrect

+ 1 pt Correct

5.4 **HasherB - Buckets** **1** / 1 pt

✓ **+ 1 pt** Correct: 2

+ 0 pts Incorrect

+ 1 pt Correct

5.5 **Incorrect** **0.5** / 1 pt

− 0 pts Correct: Hasher A and mentions ("different hash values" OR "different indexes" OR "inconsistent indexes" OR "inconsistent hash") AND ("same key" OR "same string" OR "use the key again" OR relates inconsistency/difference to keys)

✓ **− 0.5 pts** Correct hash, explanation doesn't cover the items above

− 1 pt Identifies wrong hasher

− 1 pt Blank/incorrect

Question 6

Sorting **4** / 4 pts

6.1 **A** **1** / 1 pt

✓ **− 0 pts** Correct: dosth(i, arr)

− 0.5 pts Uses 0, 1, arr.length, or arr.length-1 in first argument

− 0.5 pts Uses only a subset of arr in the second argument

− 1 pt Incorrect or blank

6.2 **B** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

6.3 **C** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

6.4 **D** **1** / 1 pt

✓ **+ 1 pt** Correct

+ 0 pts Incorrect

Question 7

Video Task

8 / 12 pts

– 0 pts Correct

– 12 pts Video can't be played on gradescope
OR
Missing any of the following:

- Id
- Audio
- Video / Video file
- Task completion

– 1 pt Video goes over 10 min, even 1 second counts (10 minutes due to typo on exam)

– 4 pts Attempts, but does not complete runtime table for **method1**

✓ – 1 pt Completes, but does not have correct number of steps in runtime table for **method1** (Even 1 mistake counts here)

– 2 pts Attempts, but does not complete runtime equation and/or total runtime for **method1**

✓ – 1 pt Completes, but does not have correct runtime equation for **method1**

– 4 pts Attempts, but does not complete runtime table for **method2**

✓ – 1 pt Completes, but does not have correct number of steps in runtime table for **method2** (Even 1 mistake counts here)

– 2 pts Attempts, but does not complete runtime equation and/or total runtime for **method2**

✓ – 1 pt Completes, but does not have correct runtime equation for **method2**

– 4 pts Bound is not right for **method2**

– 4 pts Bound is not right for **method1**

Q1 Excel with Integrity Pledge

0 Points

I will complete this exam in a fair, honest, respectful, responsible and trustworthy manner. This means that I will complete the exam as if the professor was watching my every action. I will act according to the professor's instructions, and I will neither give nor receive any aid or assistance other than what is authorized. I know that the integrity of this exam and this class is up to me, and I pledge to not take any action that would break the trust of my classmates or professor, or undermine the fairness of this class.

I promise to complete this exam in keeping with the Excel with Integrity Pledge.

Fill in your Name and today's Date below:

Andrew Onozuka 5/8/2022

Q2 Instructions
0 Points

You have until 8am on 5/8 to complete the exam. This exam is untimed. Work to maximize points. If you get stuck, work through other problems and come back to it.

In general, if you think you've spotted a typo in the exam, do your best to answer in the spirit of the question. Keep in mind that some questions have interesting code examples with intentional bugs for you to find as part of the question. **Questions asked during the exam will not be answered, so do your best to interpret the questions.**

In general, assume that any necessary libraries (JUnit, ArrayList, List, and so on) have been imported.

You can use your notes and a compiler. However, the test is designed as if you were taking it during lecture without a compiler and it's highly suggested you not rely on your compiler and use pen & paper to figure out your answers.

Stay calm – you can do this!

Q3 Partition

12 Points

Consider this specification for partition:

```
int partition(int[] numbers)
```

Partition method description:

Chooses a pivot value from the array. Then, changes the array so that all elements smaller than or equal to the pivot appear in indices less than the index of the pivot value, and all elements larger than the pivot appear in indices greater than the index of the pivot value. Returns the final index of the pivot value, which may be different than its starting index. All elements in the input should appear at some index in the output.

Q3.1 A
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{ 57, 21, 41, 59, 26, 37, 10 }
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- [✓ None of the above]

Q3.2 B
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{ 21, 37, 10, 47, 26, 57, 59 }```
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- [✓ None of the above]

Q3.3 C
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{10, 21, 26, 37, 47, 57, 59 }
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- [✓ None of the above]

Q3.4 D
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{ 37, 21, 26, 41, 59, 57, 47 }
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- [✓ None of the above]

Q3.5 E
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{ 21, 37, 59, 57, 41, 10, 26 }
```

- ☐ 0
- ☐ 1
- ☐ 2
- ☐ 3
- ☐ 4
- ☐ 5
- ☐ 6
- [✓ None of the above]

Q3.6 F
2 Points

Consider this *input* array:

```
{ 10, 57, 41, 21, 26, 37, 59 }
```

Consider the following array *after* a call to **partition()**, for the input array above. What are **all** the indices that could have been **returned** from **partition()** for this to be a valid partition result? Select **all** possible indices.

```
{10, 21, 59, 37, 26, 57, 41 }
```

- [✓ 0]
- [✓ 1]
- 2
- 3
- 4
- 5
- 6
- None of the above

Q4 Run-time

12 Points

Recall that we say *f is O(g)* if there exist *n0* and *C* such that for all *n > n0*, *f(n) < C * g(n)*

Recall that Θ (big-Theta) represents a *tight* bound, O (big-O) an *upper* bound, and Ω (big-Omega) a *lower* bound. (We intentionally do not give definitions for Θ and Ω).

Answer in terms of n

Q4.1 A

1 Point

Give a Θ bound for the number of steps the following program takes in terms of n, assuming **doSomething()** does a constant amount of work. Answer with just the function, i.e. n, n^2, x * log(n), 2^n, etc.

```
void runtime1(int n) {
    for (int i = 1; i < n; i++) {
        for (int j = 1; j <= i * i; j++) {
            if (j % i == 0) {
                for (int k = 0; k < j; k++) {
                    doSomething();
                }
            }
        }
    }
}
```

n^4

Q4.2 B
0 Points

Give a Θ bound for the number of steps the following program takes in terms of n, assuming **doSomething()** does a constant amount of work. Answer with just the function, i.e. n, n^2, x * log(n), 2^n, etc.

```
void runtime2(int n) {
  while (n > 0) {
    if (n % 3 == 0) {
      for (int i = 0; i < n * n; i += n) {
        for (int j = i; j > 0; j--) {
          doSomething();
        }
      }
    }

    n--;
  }
}
```

n^3

Q4.3 C
0 Points

Give a Θ bound for the number of steps the following program takes in terms of n, assuming **doSomething()** does a constant amount of work. Answer with just the function, i.e. n, n^2, x * log(n), 2^n, etc.

```
int runtime3(int n) {
  if (n < 1) {
    doSomething();
    return 0;
  }

  return runtime3(n-1) + runtime3(n / 2);
}
```

2^n

Q4.4 D
1 Point

Give a Θ bound for the number of steps the following program takes in terms of n, assuming **doSomething()** does a constant amount of work. Answer with just the function, i.e. n, n^2, x * log(n), 2^n, etc.

```
int runtime4(int[] n) {
    int fourths = n.length / 4;

    // Split array into fourths. Assume this always works for all n
    int[] n1 = Arrays.copyOfRange(n, 0, fourths);
    int[] n2 = Arrays.copyOfRange(n, fourths + 1, fourths * 2);
    int[] n3 = Arrays.copyOfRange(n, fourths * 2 + 1, fourths * 3);
    int[] n4 = Arrays.copyOfRange(n, fourths * 3 + 1, fourths * 4);

    for (int i = 0; i < fourths * fourths; i++) {
        doSomething();
    }

    return runtime4(n1) + runtime4(n2) + runtime4(n3) + runtime4(n4);
}
```

n^2

Q4.5 E
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = \frac{x * log(x) + 2x}{3x^2} \qquad g(x) = \frac{1}{x}$$

☐ *f is O(g)*

☑ *f is Ω(g)*

Q4.6 F
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = x! - x^{1000} \qquad g(x) = 2^x + 1$$

☐ *f is O(g)*

[✓ *f is Ω(g)*]

Q4.7 G
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = log(\ log(x)\ ) \qquad g(x) = \frac{1}{x^{10}}$$

☐ *f is O(g)*

[✓ *f is Ω(g)*]

Q4.8 H
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = (2^n)^2 \qquad g(x) = (2n)!$$

✓ *f is O(g)*

☐ *f is Ω(g)*

Q4.9 I
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = \left(\frac{\sqrt{x}}{2}\right)^3 \qquad g(x) = x^2$$

✓ *f is O(g)*

☐ *f is Ω(g)*

Q4.10 J
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = 3 * log(x) + \frac{1}{x^x} \qquad g(x) = \frac{log(x^x)}{x}$$

- [✓ *f is O(g)*]
- [✓ *f is Ω(g)*]

Q4.11 K
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = x^2 + x + \frac{1}{x^2} \qquad g(x) = \frac{(\sqrt{2x})^6}{x}$$

- [✓ *f is O(g)*]
- [✓ *f is Ω(g)*]

Q4.12 L
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = (log(x) * 2^x)^x \qquad g(x) = (x!)!$$

- [✓ *f is O(g)*]
- *f is Ω(g)*

Q4.13 M
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = \sqrt{10 * log(x) + \frac{1}{x}} \qquad g(x) = log(\frac{x}{2})$$

- [✓ *f is O(g)*]
- *f is Ω(g)*

Q4.14 N
1 Point

Consider the following pair of functions. Indicate whether *f is O(g)* and/or *f is Ω(g)*. Select zero, one, or both as appropriate.

$$f(x) = \frac{x^{10} + x^4 + 12}{\sqrt{x^4 + log(x)}} \qquad g(x) = (\frac{x^3 + \sqrt{2x}}{\sqrt{x}})^3$$

- ☐ *f is O(g)*
- [✓ *f is Ω(g)*]

Q5 HashTable

9 Points

Consider this implementation of a hash table based on one from the lecture notes, with some portions especially relevant for the question bolded.

```
interface Hasher { int hash(K key); }

class HashTable<K,V> {
  class Entry {
    K k; V v;
    public Entry(K k, V v) { this.k = k; this.v = v; }
  }
  List[] buckets; // An array of Lists of Entries
  int size;
  Hasher hasher;

  public HashTable(Hasher h, int startCapacity) {
    this.size = 0;
    this.hasher = h;
    this.buckets = (List[])(new List[startCapacity]);
  }

  public double loadFactor() { return (double)(this.size) / this.buckets.length; }

  public V get(K k) {
    int hashCode = this.hasher.hash(k);
    int index = hashCode % this.buckets.length;
    if(this.buckets[index] == null) {
      return null;
    }
    else {
      for(Entry e: this.buckets[index]) {
        if(e.k.equals(k)) { return e.v; }
      }
      return null;
    }
  }

  public void set(K k, V v) {
    int hashCode = this.hasher.hash(k);
    int index = hashCode % this.buckets.length;
    if(this.buckets[index] == null) {
      this.buckets[index] = new ArrayList();
      this.buckets[index].add(new Entry(k, v));
    }
    else {
      for(Entry e: this.buckets[index]) {
        if(e.k.equals(k)) { e.v = v; return; }
      }
      this.buckets[index].add(new Entry(k, v));
    }
    this.size += 1;
  }

}
```

Consider these implementations of the **Hasher** interface:

```
class HasherA implements Hasher<String> {
  int i;
  public HasherA() { this.i = 0; }
  public int hash(String s) { i += 1; return s.length() + i; }
}

class HasherB implements Hasher<String> {
  public int hash(String s) { return Character.codePointAt(s, 0); }
}
```

Recall that Character.codePointAt retrieves the ASCII code of the character at the given index. Consider this sequence of operations:

```java
HashTable<String, Integer> ht = new HashTable<>(???, 6);
ht.set("first", 100);
System.out.println(ht.size);
ht.set("second", 100);
System.out.println(ht.size);
System.out.println(ht.get("second"));
ht.set("first", 300);
System.out.println(ht.size);
System.out.println(ht.get("first"));
ht.set("fourth", 500);
System.out.println(ht.size);
```

Consider using each of the two Hasher implementations to replace the **???** above by filling it in with either **new HasherA()** or **new HasherB()**.

ASCII Table:

a	97	f	102	k	107	p	112	u	117	z	122
b	98	g	103	l	108	q	113	v	118		
c	99	h	104	m	109	r	114	w	119		
d	100	i	105	n	110	s	115	x	120		
e	101	j	106	o	111	t	116	y	121		

Q5.1 HasherA - Output
3 Points

For HasherA, write the **output** below (6 lines of output):

```
1
2
null
3
null
4
```

Q5.2 HasherB - Output
3 Points

For HasherB, write the **output** below (6 lines of output):

```
1
2
100
2
300
3
```

Q5.3 HasherA - Buckets
1 Point

For HasherA, at the end of running all the statements, how many buckets would be non-empty? Write your answer as a number below:

3

Q5.4 HasherB - Buckets
1 Point

For HasherB, at the end of running all the statements, how many buckets would be non-empty? Write your answer as a number below:

2

Q5.5 Incorrect
1 Point

Which of the two produces *incorrect* results? Identify which one does, and **give a one-sentence answer why**:

HasherA produces the incorrect results because i is incremented when calling hash and not set to 0.

Q6 Sorting
4 Points

Consider this helper method:

```
public static void dosth(int loc, int[] arr) {
    int j = loc;
    while (j > 0 && arr[j] < arr[j-1]){
      int temp = arr[j];
      arr[j] = arr[j-1];
      arr[j-1] = temp;
      j--;
    }
  }
```

Q6.1 A
1 Point

Consider this method skeleton:

```
void isort(int[] arr) {
  for(int i = 0; i < arr.length; i += 1) {

  }
}
```

Fill in a **single call to the helper method dosth()** that would complete the method and cause it to always change the input array to be in **increasing sorted order**. Write the body of the loop (a single call to dosth):

```
dosth(i, arr);
```

Q6.2 B
1 Point

Give a Θ bound (a tight bound) for the number of steps the **dosth()** method takes in terms of n in the **worst case** arrangement of the elements in arr, and assuming arr.length is at least as large as n.

- Θ(1)
- Θ(log(n))
- [✓ Θ(n)]
- Θ(n*log(n))
- Θ(n^2)
- Θ(n^3)
- Θ(2^n)
- Θ(n!)

Q6.3 C
1 Point

Give a Θ bound (a tight bound) for the number of steps the **dosth()** method takes in terms of n in the **best case** arrangement of the elements in arr, and assuming arr.length is at least as large as n.

- [✓ Θ(1)]
- Θ(log(n))
- Θ(n)
- Θ(n*log(n))
- Θ(n^2)
- Θ(n^3)
- Θ(2^n)
- Θ(n!)

Q6.4 D
1 Point

Give a Θ bound (a tight bound) for the number of steps the completed **isort()** method you wrote takes in terms of arr.length in the **worst case** arrangement of the elements in arr, for input arrays of arbitrary size.

- Θ(1)
- Θ(log(n))
- Θ(n)
- Θ(n*log(n))
- [✓ Θ(n^2)]
- Θ(n^3)
- Θ(2^n)
- Θ(n!)

Q7 Video Task

12 Points

You will record and upload a video of no more than 6 min.

- Show only your face and a picture ID (your student ID is preferred but any picture ID with your name on it will work) for a few seconds at the beginning. You don't have to be on camera the whole time, though it's fine if you are. Just a brief confirmation that it's you creating the video/doing the work attached to the work itself is what we want. If you do not have a webcam, take a picture of yourself (and your picture ID) with your phone and display that picture at the start of your screen share.
- You must also verbally explain your thought process as you complete the following task.

Task:

Download the code file and fill in the table and following blanks:

https://github.com/ucsd-cse12-sp22/cse12-exam2-Q7/blob/main/Exam2_Video.java

---

When you are complete, upload your video to this question.

Note: Gradescope has a max file size of 100MB. Any files larger than this will be rejected. We will **NOT** accept any files (code, videos, etc.) outside of the gradescope submission. Additionally make sure that you are able to play your video on gradescope. If your video is not playable on gradescope by staff, then it will not be graded and given a score of 0.

Tips for reducing video file sizes:

- Use zoom
  - A 100MB video is about a 20 minute video on Zoom (1620x1080 video resolution).
  - If you are not using Zoom, convert your file to an MP4 (there are web tools that can do this). MP4 videos have smaller video sizes than other video formats.
- Use a smaller screen size for your computer, or only record a smaller portion of the screen.
- Reduce background clutter (i.e. desktop icons). Background clutter reduces compression, making larger file sizes. You can hide the background by maximizing your code editor window.
- Keep your video short.
  - 10 minutes is the max, it's possible to cover everything in less time
  - Create a script of what you are going to say.
  - Pause the video as you switch to the code for the next question (make sure to tell us which question you are showing the code for when you switch).
- Use a 3rd party video editor to reduce the width/height of the video, but make sure your code isn't blurry when you play it on Gradescope as we cannot grade blurry videos.

Exam2Video.mp4 Download