Final

Graded

Student

Colin Cano

Total Points

22 / 40 pts

Question 1

Q **4** / 5 pts

- **+ 0 pts** The options below are in **alternative** to one another
- **+ 0 pts** All answers are incorrect
- **+ 1 pt** One answer is correct
- **+ 2 pts** Two answers are correct
- **+ 3 pts** Three answers are correct
- ✔ **+ 4 pts** Four answers are correct
- **+ 5 pts** All answers are correct

Question 2

Q **0** / 2 pts

- ✔ **+ 0 pts** The options below are **cumulative**.
- **+ 1 pt** myPrint(last) is correct
- **+ 1 pt** myPrint(first) is correct

Question 3

Q **0** / 1 pt

- ✔ **+ 0 pts** Incorrect
- **+ 1 pt** Correct

Question 4

Q **2** / 2 pts

- **+ 0 pts** The options below are in **alternative** to one another
- **+ 0 pts** Incorrect
- **+ 1 pt** Answer is $O(n^2)$
- ✔ **+ 2 pts** Answer is $O(n \log n)$

Question 5

Q **2** / 2 pts

+ 0 pts The options below are **cumulative**.

✔ **+ 1 pt** 1) is correct.

✔ **+ 1 pt** 2) is correct.

Question 6

Q **0** / 1 pt

✔ **+ 0 pts** Incorrect

+ 1 pt Correct

Question 7

Q **0** / 2 pts

✔ **+ 0 pts** Incorrect

+ 2 pts Correct

Question 8

Q **2** / 2 pts

+ 0 pts Incorrect

✔ **+ 2 pts** Correct

Question 9

Q **0** / 2 pts

✔ **+ 0 pts** Incorrect

+ 2 pts Correct

Question 10

Q **5** / 5 pts

10.1 **Post-order** **1** / 1 pt

+ 0 pts Incorrect

✔ **+ 1 pt** Correct

10.2 **In-order** **1** / 1 pt

+ 0 pts Incorrect

✔ **+ 1 pt** Correct

10.3 **Breadth-first** **1** / 1 pt

+ 0 pts Incorrect

✔ **+ 1 pt** Correct

10.4 **Frontier** **2** / 2 pts

+ 0 pts The options below are in **alternative** to one another

+ 0 pts Incorrect

+ 1 pt List is in reverse order

✔ **+ 2 pts** Correct

Question 11

Q **2** / 3 pts

11.1 **remove 46** **0** / 1 pt

✔ **+ 0 pts** Incorrect

+ 1 pt Correct

11.2 **remove 89** **1** / 1 pt

+ 0 pts Incorrect

✔ **+ 1 pt** Correct

11.3 **remove root** **1** / 1 pt

+ 0 pts Incorrect

✔ **+ 1 pt** Correct

Question 12

Q **2** / 4 pts

+ 0 pts The options below are **cumulative**.

✔ **+ 1 pt** 1) is correct

+ 1 pt 2) is correct

✔ **+ 1 pt** 3) is correct

+ 1 pt 4) is correct

Question 13

Q **2** / 2 pts

+ 0 pts The options below are in **alternative** to one another

✔ **+ 0.5 pts** (a) is correct

✔ **+ 0.5 pts** (b) is correct

✔ **+ 0.5 pts** (c) is correct

✔ **+ 0.5 pts** (d) is correct

+ 0 pts None correct.

Question 14

Q **0** / 2 pts

✔ **+ 0 pts** Incorrect

+ 2 pts Correct

Question 15

Q **1** / 2 pts

+ 0 pts Incorrect

✔ **+ 1 pt** Partially correct

+ 2 pts Correct

Question 16

Q **0** / 3 pts

✔ **+ 0 pts** Incorrect

+ 3 pts Correct

Name (First and Last): Colin Camp

CS2230 Computer Science II: Data Structures

Final Exam

May 14, 2025

You may have 3 pages front and back of notes, written and/or printed. No electronics, no calculators.

For multiple choice questions, you can circle your choice or put a check mark on it, as you prefer, but make sure your mark is clear.

We will be scanning the exams before grading them, so to help us out please *write reasonably dark.* Either pencil or pen is fine.

This booklet should have **10 pages**, including this one. *Notify the proctors immediately if not.*

Take a deep breath ... and relax

Grading

This exam assesses your individual level of mastery of the concepts, ***not*** your ranking among students. Everyone has a chance to earn the grade they desire.

Question 0

a) If the following statement is true, then write "I did not give or receive help in the taking of this exam" and sign your name.

Do not open the exam before being given permission

Name (First and Last): Colin Cano

1. (5 points)

Consider a Java implementation of a doubly linked list that maintains a reference to the first and to the last node in the list, along with its size, as we have seen in this course.

```
public class DoublyLinkedList<E> {
  private Node first; // first node in the list
  private Node last;  // last node in the list
  private int size;   // number of items in the list

  private class Node {
    private Item elem; // the list element
    private Node next; // next node
    private Node prev; // previous node
  }
  ...
}
```

What are the best estimates of the **worst-case** runtimes of the following operations in big-O notation?

Choose among $O(1), O(\log n), O(n), O(n \log n)$ where n is the value of field `size`.

		Answer
`addFirst(e)`	Add `e` to the beginning of the list	O(1)
`get(i)`	Return the element at position `i` in the list	O(N)
`set(i, e)`	Replace element at position `i` of the list with `e`	O(nlogn)
`removeLast()`	Delete and return the element at the end of the list	O(1)
`contains(e)`	Return true if and only if `e` is in the list	O(N)

2. (2 points)

Consider the class `DoublyLinkedList` in the previous question and suppose it has the following method:

```
void myPrint(Node<T> n) {
  if (n == null) return;
  myPrint(n.prev);
  System.out.print(n.elem);
}
```

In an instance of `DoublyLinkedList<Integer>` that contains, in order, the exactly the elements: 1, 2, 3, 4, 5 :

1. what would the call `myPrint(last)` print? **Answer:** nothing
2. what would the call `myPrint(first)` print? **Answer:** nothing

In each answer, write the word "nothing" if the program does not print anything.

1 2 3 5 6 7 8 Name (First and Last): Colin Ceno 2 1 3 6 8 7

3. (1 point)

The height of a Binary Search Tree (BST) depends on the order in which the keys are inserted into an empty tree *if no balancing operation* is performed.

Given an initially empty BST, in what temporal order would you insert the keys 1, 2, 3, 5, 6, 7, 8 so as to minimize the height of the BST? (Assume that the keys are compared by the usual < relation.)

Answer: 5, 1, 8, 2, 3, 6, 7

4. (2 points)

What is the best estimate of the **worst-case** time complexity of the following Java method using the big-O notation?

```
public static void f(int n) {
  int sum = 0;
  for (int i = 1; i <= n; i = 2*i)
    for (int j = 0; j <= n; j++)
      sum = sum + 1
}
```

O(logn)
O(n)

Answer: O(nlogn)

5. (2 points)

Consider the following Java interfaces and classes. The bodies are omitted, but you can assume that the code compiles successfully.

```
interface A { ... }
interface B extends A { ... }
class C implements B { ... }
abstract class D implements B { ... }
```

1) Here are some statements about class C. Mark all (and only) the true ones.
   - (a) C can optionally implement any number of methods declared in A
   - (b) C must implement all methods declared in A
   - (c) C can optionally implement any number of methods declared in B
   - (d) C must implement all methods declared in B

2) Here are some statements about D. Mark all (and only) the true ones
   - (a) D can optionally implement any number of methods declared in A
   - (b) D must implement all methods declared in A
   - (c) D can optionally implement any number of methods declared in B
   - (d) D must implement all methods declared in B

Name (First and Last): Colin Cano

6. (1 point)

My application needs a `List<Integer>` instance and has the following behavior:

- frequently searches if an given integer is in the list;
- very infrequently adds an integer to the list.

Which `List` implementation should I use to best suit this application?
Mark only one. Choose (e)-(g) when two options are best, asymptotically.

(a) `LinkedList` and leave elements unsorted
(b) `LinkedList` and keep elements sorted
(c) `ArrayList` and leave elements unsorted
(d) `ArrayList` and keep elements sorted
(e) option (a) or (b)
(f) option (c) or (d)
(g) option (b) or (d)

7. (2 points)

Which of the following is a possible interaction with a correctly implemented `Iterator<Integer>` in Java?

(a) `hasNext()` returns `false`, then `next()` returns `null`
(b) `hasNext()` returns `false`, then `next()` returns `null`, then `hasNext()` returns `true`
(c) `hasNext()` returns `false`, then `hasNext()` returns `true`, then `next()` returns `7`
(d) `hasNext()` returns `true`, then `hasNext()` returns `false`
(e) `hasNext()` returns `true`, then `hasNext()` returns `true`, then `next()` returns `null`
(f) `hasNext()` returns `false`, then `next()` returns `42`
(g) `hasNext()` returns `true`, then `hasNext()` returns `true`, then `hasNext()` throws `NoSuchElementException`

8. (2 points)

With regard to running time, pick the most accurate definition of ***best case*** versus ***worst case***.

(a) The best case is small inputs which give the shortest running time and the worst case is large inputs which give the longest running time.
(b) The best case is the algorithm with the shortest running time for the problem and the worst case is the algorithm with the longest running time for the problem.
(c) The best case is the lower bound running time which we express with Ω and the worst case is the upper bound running time with we express with O.
(d) There is no distinction between best case and worst case because asymptotic analysis specifically deals with large inputs.
(e) The best case is the input that gives the shortest running time and the worst case is the input (of the same size) that gives the longest running time.
(f) The best case is inputs for which the algorithm only needs to examine a constant number of elements, while worst case is inputs for which the algorithm must examine every element.
(g) If I guess, best case I get this question right, worst case I do not.

Name (First and Last): Colin Ceno

9. (2 points)

Consider this method of some Java class generic in the type K and V.

```
10  K foo(Position<Entry<K,V>> p, K x) {
11    this.root = p;
12    if (root != null)
13      return root.getElement().getKey();
14    Entry<K,V> e1 = new MapEntry(x, null);
15    Entry<K,V> e2 = e1;
16    return e2.getKey();
17  }
```

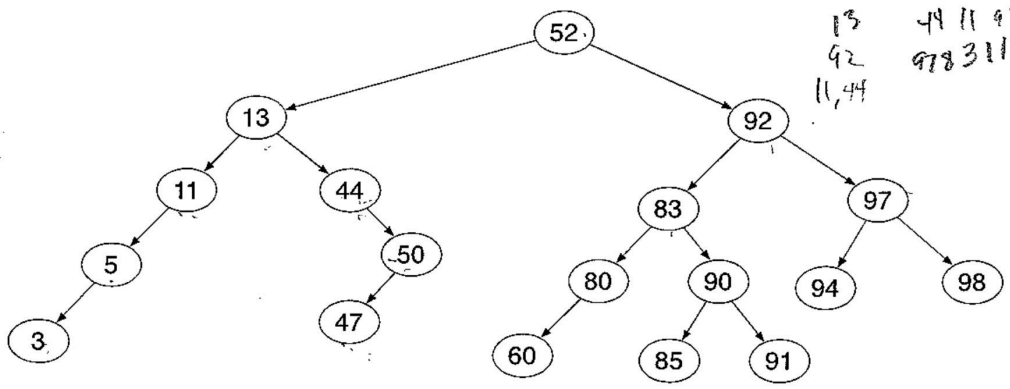
When we ran our code, we got a `NullPointerException` on one of these lines. Only one line could have caused such an error. Which one?

Answer: 15

10. (5 points)

In this question you will identify the depth-first and breadth-first traversals for this **binary tree**.

52 92 13 44 11 92
13 44 11 92 44
92 97 83 11 44
11, 44



Consider the following traversals of the tree above:

(a) 52, 13, 92, 11, 44, 83, 97, 5, 50, 80, 90, 94, 98, 3, 47, 60, 85, 91
(b) 52, 13, 92, 11, 44, 5, 50, 3, 47, 83, 97, 80, 90, 94, 98, 60, 85, 91
(c) 3, 5, 11, 47, 50, 44, 13, 60, 80, 85, 91, 90, 83, 94, 98, 97, 92, 52
(d) 3, 47, 60, 85, 91, 5, 50, 80, 90, 94, 98, 11, 44, 83, 97, 13, 92, 52
(e) 52, 13, 11, 5, 3, 92, 97, 98, 44, 50, 47, 83, 80, 60, 90, 91, 94, 85
(f) 3, 5, 11, 13, 44, 47, 50, 52, 60, 80, 83, 85, 90, 91, 92, 94, 97, 98
(g) 52, 13, 11, 5, 3, 44, 50, 47, 92, 83, 80, 60, 90, 85, 91, 97, 94, 98

1) Which one is the depth-first **post-order** traversal? (Mark one)

(a) (b) (c) (d) (e) (f) (g)

Name (First and Last): Colin Cano

2) Which one is the depth-first **in-order** traversal? (Mark one)

(a) (b) (c) (d) (e) (f) (g)

3) Which one is the **breadth-first** traversal? (Mark one)

(a) (b) (c) (d) (e) (f) (g)

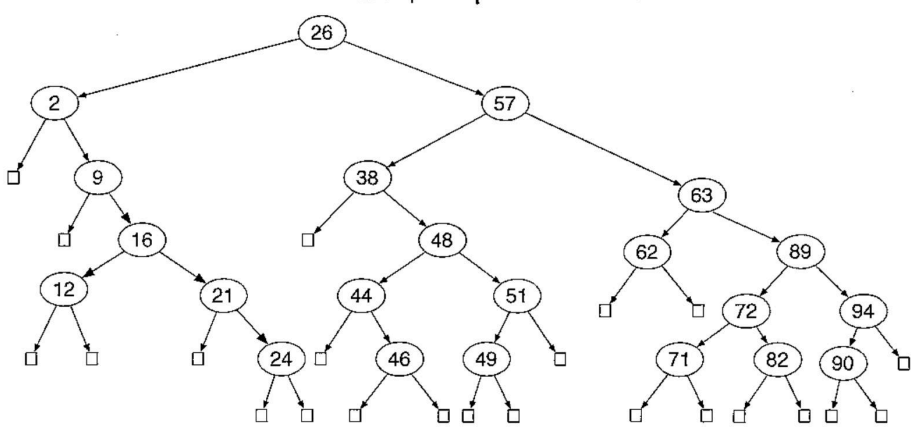
4) (2 points) If you were doing a **breadth-first** traversal of the tree above using a loop, what would be the contents of the fringe (or, frontier) **right before** the node with value 11 is visited?
Assume that when expanding a node its left child is added to the fringe its right child. Write the fringe as a list where the leftmost value is the most recent one.

Answer: [97, 83, 44, 11]

11. (3 points)

For the following **binary search tree** consider the questions below *independently*. Operations on the tree are performed as described in the lecture nodes and the textbook code we studied in class.

24, 21, 16, 12, 9, 2



1) What will be the right child of (of the node with value) 44 after we remove 46?

(a) 38
(b) 48
(c) 51
(d) 49
(e) 57
(f) an external node
(g) null

2) What best describes how the subtree currently rooted at 63 will look **after** removing 89?

(a) 63's new right child is null
(b) 63's new right child is an external node
(c) 63's new right child is 94. 94 has two children: 72 on the left and 90 on the right
(d) 63's new right child is 82. 82 has two children: 72 on the left and 94 on the right
(e) 63's new right child is 72. 72 keeps its children and 94 becomes the right child of 82
(f) 82 goes where 63 is. 82 has two children: 63 on the left and 94 on the right
(g) 89 cannot be removed because it has two children

Name (First and Last): Colin Cano

3) How many times must we call remove on the current root of the tree before 2 is the root?

Answer: 6

12. (4 points)

Refer to the code and box-and-arrow diagram on the last page of the exam **Map ADT code**. The code has an implementation of the Map ADT. *Study the code carefully before answering the questions below.*

1) Suppose we start with an empty `AppendMap m`. Which of the method call sequences below would produce the given box-and-arrow diagram? (Mark one)

   (a) `m.put(7, 88); m.put(22, 88); m.put(15, 42); m.put(15, 96); m.put(7, 70);`

   (b) `m.put(7, 88); m.put(22, 88); m.put(15, 42); m.put(15, 96); m.put(7, 70); m.put(7, 70);`

   (c) `m.put(7, 70); m.put(15, 96); m.put(15, 42); m.put(22, 88); m.put(7, 88);`

   (d) `m.put(7, 70); m.put(15, 96); m.put(15, 42); m.put(22, 88); m.put(15, 42); m.put(7, 88);`

   (e) `m.put(15, 42); m.put(22, 88); m.put(7, 88);`

2) What will be returned by `m.size()`? **Answer:** 7

3) What will be returned by `m.entries.size()`? **Answer:** 5

4) Which of following mapping sets represents the contents of `Map m`? (Mark one)

   (a) $\{7 \mapsto [70, 88],\ 15 \mapsto [96, 42],\ 22 \mapsto [88]\}$

   (b) $\{7 \mapsto 88, 15 \mapsto 42\}$

   (c) $\{7 \mapsto 88, 15 \mapsto 42, 22 \mapsto 88\}$

   (d) $\{7 \mapsto 70, 15 \mapsto 96, 22 \mapsto 88\}$

   (e) $\{7 \mapsto 70, 15 \mapsto 96, 15 \mapsto 42, 22 \mapsto 88\}$

   (f) $\{7 \mapsto 70, 15 \mapsto 96, 15 \mapsto 42, 7 \mapsto 88\}$

   (g) $\{7 \mapsto 70, 15 \mapsto 96, 15 \mapsto 42, 22 \mapsto 88, 7 \mapsto 88\}$

13. (2 points)

Consider the following *iterator transformers*:

- `Iterator<Integer> Twice(Iterator<Integer> it)`
  takes as input an iterator generating a sequence $[d_0, d_1, d_2, \ldots]$ of `Integer` values and returns an iterator generating the sequence $[2d_0, 2d_1, 2d_2, \ldots]$ of their doubles (e.g., given an iterator generating the sequence [2, 4, 1, …] it returns an iterator generating [4, 8, 2, …]).

Name (First and Last): Colin Cano

- `Iterator<Integer> Count(Iterator<Integer> it)`
  takes as input an iterator generating a sequence $[d_0, d_1, d_2, \ldots]$ of `Integer` values and returns an iterator generating the sequence $[1, 2, 3, \ldots]$.

- `Iterator<Integer> Sum(Iterator<Integer> it)`
  takes as input an iterator generating a sequence $[d_0, d_1, d_2, \ldots]$ of `Integer` values and returns an iterator generating the sequence $[d_0,\ d_0 + d_1,\ d_0 + d_1 + d_2, \ldots]$.

Each of the following expressions returns an iterator. Write down the sequence generated by the returned iterator, where `iter0` is a finite iterator generating the sequence [1, 1, 5]. [1, 2, 7] [1, 2, 7] [1, 2, 3]

(a) `Twice(Sum(iter0))` **Answer:** [2, 4, 14]

(b) `Twice(Count(Sum(iter0)))` **Answer:** [2, 4, 6]

(c) `Sum(Twice(Count(iter0)))` **Answer:** [2, 6, 12]

(d) `Count(Twice(Sum(iter0)))` **Answer:** [1, 2, 3]

14. (2 points)

Suppose you have a binary tree with n nodes. You perform a breadth-first traversal but stop as soon as you encounter a leaf node. What are the best case and worst case running times?

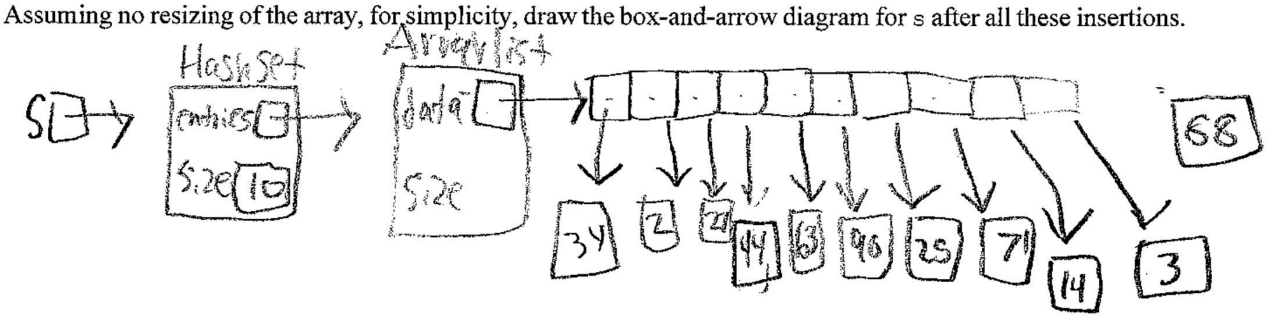
(a) $\Theta(1)$, $\Theta(\log_2 n)$
(b) $\Theta(1)$, $\Theta(n \log_2 n)$
(c) $\Theta(1)$, $\Theta(n)$
(d) $\Theta(1)$, $\Theta(n^2)$
(e) $\Theta(\log_2 n)$, $\Theta(n)$
(f) $\Theta(n)$, $\Theta(n)$
(g) $\Theta(n)$, $\Theta(n^2)$

15. (2 points)

Let `s` be an instance of `HashSet<Integer>` implemented with an array of **size 10** and based on chaining to deal with collisions. Suppose that after creating `s` we insert the following values into, in this order:

34, 2, 27, 44, 63, 90, 25, 71, 14, 88, 3.

Assuming no resizing of the array, for simplicity, draw the box-and-arrow diagram for `s` after all these insertions.



Name (First and Last): Colin Ceno

16. Optional, Extra Credit (3 points)

Consider the following Java code for a binary tree.

```
public interface Position <E> {
  E getElement ();
}
public interface Tree <E> extends Iterable <E> {
  Position <E> root();
  Position <E> parent(Position <E> p);
  Iterable <Position <E>> children(Position <E> p);
  int numChildren(Position <E> p);
  boolean isInternal(Position <E> p);
  boolean isExternal(Position <E> p);
  boolean isRoot(Position <E> p);
  int size ();
  boolean isEmpty ();
  Iterator<E> iterator();
  Iterable<Position<E>> positions();
}

class MyTree <E> implements Tree <E> {
  public int depth(Position<E> p) { … }
  public int height(Position<E> p) { … }

  public Position <E> foo(Position <E> a, Position <E> b) {
   if (a == b)
     return a;
   else if (this.depth(a) >= this.depth(b)) {
     return foo(this.parent(a), b);
   else
     return foo(a, this.parent(b));
  }
}
```

Suppose, `t` is an instance of `MyTree` and `p1` and `p2` are two positions (nodes) in `t`. What does the call `t.foo(p1,p2)` return?

(a) the root of `t`
(b) the common child between `p1` and `p2`, if any; **null** otherwise
(c) a leftmost node that is deeper in the tree `t` than `p1` and `p2`
(d) the deepest node in `t` that is an ancestor of both `p1` and `p2`
(e) the parent of `p1` and `p2` if they are siblings; **null** otherwise
(f) `p1` if it is an ancestor of `p2`; otherwise, `p2`
(g) `p1` if its depth is smaller than `p2`'s depth; otherwise, `p2`