Midterm 2 (raw points)

Graded

Student

Colin Cano

Total Points

19.81 / 41 pts

Question 1

Problem 1 **10.81** / 18 pts

1.1 **Question 1** **3.66** / 6 pts

A description of the worst case input.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- **+ 0.33 pts** mostly incorrect
- **+ 0.66 pts** mostly correct but not completely
- ✔ **+ 1 pt** fully correct

An initial summation used to calculate the steps.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- **+ 0.66 pts** mostly incorrect
- ✔ **+ 1.33 pts** mostly correct but not completely
- **+ 2 pts** Fully correct

Explanation of how the code relates to the summation.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- **+ 0.66 pts** mostly incorrect or incomplete
- ✔ **+ 1.33 pts** mostly correct and complete but not fully
- **+ 2 pts** Fully correct and complete

A Θ bound for the worst-case scenario.

Choices are **additive**.

- **+ 0.5 pts** Correct bound
- **+ 0.5 pts** Sufficiently detailed and correct explanation

1.2 **Question 2** **2.99** / 6 pts

A description of the worst case input.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- **+ 0.33 pts** mostly incorrect
- **+ 0.66 pts** mostly correct but not completely
- ✔ **+ 1 pt** fully correct

An initial summation used to calculate the steps.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- ✔ **+ 0.66 pts** mostly incorrect
- **+ 1.33 pts** mostly correct but not completely
- **+ 2 pts** Fully correct

Explanation of how the code relates to the summation.

Choices are **in alternative**.

- **+ 0 pts** blank or nonsense
- **+ 0.66 pts** mostly incorrect or incomplete
- ✔ **+ 1.33 pts** mostly correct and complete but not fully
- **+ 2 pts** Fully correct and complete

A Θ bound for the worst-case scenario.

Choices are **additive**.

- **+ 0.5 pts** Correct bound
- **+ 0.5 pts** Sufficiently detailed and correct explanation

1.3 **Question 3** **4.16** / 6 pts

A description of the worst case input.

Choices are **in alternative**.

+ 0 pts blank or nonsense

✔ **+ 0.33 pts** mostly incorrect

+ 0.66 pts mostly correct but not completely

+ 1 pt fully correct

An initial summation used to calculate the steps.

Choices are **in alternative**.

+ 0 pts blank or nonsense

+ 0.66 pts mostly incorrect

✔ **+ 1.33 pts** mostly correct but not completely

+ 2 pts Fully correct

Explanation of how the code relates to the summation.

Choices are **in alternative**.

+ 0 pts blank or nonsense

+ 0.66 pts mostly incorrect or incomplete

+ 1.33 pts mostly correct and complete but not fully

✔ **+ 2 pts** Fully correct and complete

A Θ bound for the worst-case scenario.

Choices are **additive**.

✔ **+ 0.5 pts** Correct bound

+ 0.5 pts Sufficiently detailed and correct explanation

Question 2

Problem 2 **5.5** / 12 pts

2.1 **1.** **1** / 4 pts

+ **0 pts** Choices are **additive**

✔ + **1 pt** Provides abstract view (no concrete details of the implementation)

+ **1.5 pts** Correctly identifies top element

+ **1.5 pts** Abstract view, if provided, is correct (repeated elements are properly accounted for).

+ **0 pts** No answer or an improper answer.

2.2 **2.** **2.5** / 4 pts

+ **0 pts** Choices are **additive**

+ **1 pt** Declares and initializes s correctly

✔ + **1 pt** Has right idea of calling s.push on the data that need to be in the stack

✔ + **1 pt** Has the right number of pushes (7)

+ **1 pt** Pushes are in the right chronological order (from 50 to 30).

+ **0 pts** No answer or improper answer.

✔ + **0.5 pts** Initializes s mostly properly.

2.3 **3.** **2** / 4 pts

+ **0 pts** Choices are **additive**

✔ + **1 pt** Calls methods on s

✔ + **1 pt** Uses pop and uses it properly

+ **2 pts** Correctly leads to just two Node instances

+ **0 pts** No answer or improper answer.

+ **0.5 pts** Calls pop mostly correctly.

Question 3

Problem 3

3.5 / 8 pts

Fields and constructors.

Choices are **additive**

- ✔ **+ 0.5 pts** Declares fields
- ✔ **+ 0.5 pts** All fields are private
  - **+ 1 pt** Field choice and type are not overly complicated (an int counter suffices)
- ✔ **+ 0.5 pts** Properly initializes the object

---

hasNext

Choices are **additive**

- ✔ **+ 1 pt** checks correctly that there is another element
  - **+ 0.5 pts** the check is not overly complicated

---

next

Choices are **additive**

- ✔ **+ 1 pt** Throws exception if there is no next element
  - **+ 1 pt** saves current element before updating object state
  - **+ 1 pt** updates object states for next time
  - **+ 1 pt** returns current element

- **+ 0 pts** ****completely Incorrect.

Question 4

Problem 4

0 / 3 pts

- ✔ **+ 0 pts** Choices are **additive**
  - **+ 0.5 pts** Correct understanding that the runtime add depends on the number n of elements in the list
  - **+ 0.5 pts** Correct understanding that, after i additions, the runtime of add is actually proportionally to $n + i$
  - **+ 0.5 pts** correct and tight summation
  - **+ 0.5 pts** enough details of the calculation
  - **+ 1 pt** Correct result

Name (First and Last): Colin Cano

CS2230 Computer Science II: Data Structures

Midterm Exam II

April 8, 2025

You may have 2 pages front and back of notes, written and/or printed. Nothing else. No electronics, no calculators. **Show your work** when appropriate.

We will be scanning the exams before grading them, so to help us out please write reasonably dark. Either pencil or pen is fine. ***The back of these pages as for scratch work only***, *they will not be scanned.*

You can work on the problems in any order. If you are stuck on a problem, move to another one and go back to the previous problem later.

This booklet should have 10 pages, including this one. *Let the proctors know immediately if it does not.*

Grading

Problem	Points
1	18
2	12
3	8
Extra	3
Total	38

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 aspire to. Letter grades will be assigned as specified in the syllabus based on your normalized total score (out of 100).

0. Statement

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. I did not give or receive help in the taking of this exam.

Colin Cano

Do not start the exam until you are told to do so

Name (First and Last): Colin Cano

1. Analysis of algorithms (18 points)

For each of the three problems below, provide the following, clearly labeled as *(i), (ii), (iii), (iv)*.

i. **(1 pt)** A description of the worst-case input (the one that would cause the largest runtime).

ii. **(2 pt)** An initial summation used to calculate the number of steps for the worst case.

iii. **(2 pt)** An explanation of how the code relates to the summation

   a) (loops require discussing number of iterations and steps per iteration)
   b) (recursion requires a diagram).

iv. **(1 pt)** A Θ bound for the worst case scenario. *Show how you simplified the summation in (ii)* to find the bound.

[Go to next page.]

Name (First and Last): Colin Cano

Question 1 What is the runtime of `fun1` when called on an array `a` of length n? You can assume for simplicity that n is a power of 2.

```
static void fun1(int[] a) {
  for (int k = 1; k < a.length; k = 2 * k) {
    for (int j = 0; j < a.length; j++) {
      if (a[j] < 0) { break; }
      else { a[j] = a[j] + 3; }
    }
    for (int i = 0; i < k; i++) {
      a[k] = a[k] * 2;
      a[i] = a[k] * a[k];
    }
  }
}
```

i.) the worst case would be an array with every integer > 0

ii) $\sum_{k=1}^{a.length-1} \left(\sum_{j=0}^{a.length-1} 1 + \sum_{i=0}^{k-1} 1 \right)$

iii) the K loop has a.length-1 iterations, as does the j loop. a[j] = a[j]+3; is only true at most once per iteration. The body of the j loop is O(1). In the worst-case all of the loops would run which all take O(1) steps.

iiii). $\Theta(n^2)$

Name (First and Last): Colin Cano

Question 2 What is the runtime of `fun2`? You can assume, for simplicity, that the input value n for `fun2` is a power of 3.

```
static void fun2(int n) { fun2Aux(n, n * n); }

private void fun2Aux(int a, int b) {
  if (a <= 1)
    System.out.println("done");
  else
    for (int i = 0; i < b; i++)
      System.out.println("working");
  fun2Aux(a/3, b-1);
}
```

i.) worst case is when a > 1

ii) $\sum_{i=0}^{n/3} (n - n^2 + 1)$

iii)

fun2(n) — n

fun2Aux($\frac{n}{3}$, n^2-1) — $\frac{n}{3}$

fun2Aux($\frac{n}{9}$, n^2-2) — $\frac{n}{9}$

...

fun2aux(1) — 1

each recursion for a is divided by 3 until it reaches 1

iv.) $\Theta(n)$

Name (First and Last): Colin Como

Question 3 What is the runtime of `fun3` when called on a `LinkedList` of size n? Assume an implementation of `LinkedList` that stores internally references to the head and the tail of the list *but not its size*. As usual, method `add` adds its input element at the end of the list.

```
static void <T> fun3(LinkedList<T> ls) {
  int n = ls.size();
  for (int i = 0; i < n; i++) {    O(n)
    T e = ls.get(i);    O(n)
    ls.add(e);
  }

}
```

i.) Worst case is when $n > 0$

ii.) $\sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} n \right)$

iii.) In LinkedList, ls.get is O(n). In each iteration, i is incremented by one so there will n-1 iterations.

iiii.) $\Theta(n^2)$

Name (First and Last): Colin Cano

2. Abstract Data Types and Data Structures (12 points)

On the last two sheets of this booklet, you can find the code for the `Stack` interface, the `CompStack` class, and a related box-and-arrow diagram.

1. (4 points) Given the box-and-arrow diagram, what would be the corresponding **abstract view** (a sequence of elements) of the `Stack` instance pointed to by `s`? That is, what are its contents? *Make sure to indicate which of the elements is at the top of the stack.*

[30, 50, 70, 90]
top

2. (4 points) Provide an example of Java code that would produce the given box-and-arrow diagram.

```
private StackCE7 s = new StackCE7;
s.push(30);
s.push(30);
s.push(70);
s.push(90);
s.push(70);
s.push(70);
s.push(30);
```

3. (4 points) Assuming the given box-and-arrow diagram as a starting point, what additional Java code would lead to the diagram having only **two** `Node` objects remaining connected to the data structure?

```
s.pop();
s.pop();
```

Name (First and Last): Colin Cano

3. Iterators (8 points)

Prove an implementation of iterator `CountDown` which generates a countdown to 0 (inclusive) from an integer n give to its constructor as input. For example, `CountDown(5)` generate the sequence 5, 4, 3, 2, 1, 0. In contrast, `CountDown(-3)` generates the empty sequence since it is not possible to go down from -3 to 0.

```
public class CountDown implements Iterator<Integer> {
  // add instance variables as needed
   private Iterator<Integer> inp;
   private int current;
   private int end;
   public CountDown (int n) {
      this.end = n;
      this.current = 0

   }

   public boolean hasNext() {
        return current < end;



   }

   public Integer next() {
      if (!hasNext()){ throw new NoSuchElementException();}
      current++;
      return inp.next();


   }
}
```

Name (First and Last): Colin Cano

4. Optional, extra credit (3 points)

Redo the analysis for `fun3` in Problem 1 assuming this time that the instance stores internally only a reference to the head of the list but not to the tail.

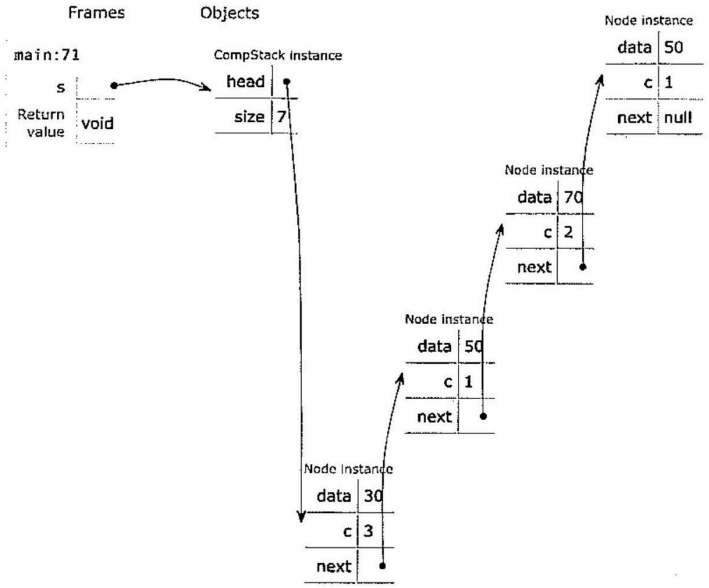
Name (First and Last): Colin Cano

Addendum

For convenience, *you can detach this sheet and the next, but you must return both sheets with the rest of the exam.*

```
/*
 * Copyright 2014, Michael T. Goodrich, Roberto Tamassia, Michael H. Goldwasser
 */
/**
 * A collection of objects that are inserted and removed according to the last-in
 * first-out principle.
 */
public interface Stack<E> {
  /**
   * Returns the number of elements in the stack.
   * @return number of elements in the stack
   */
  int size();

  /**
   * Tests whether the stack is empty.
   * @return true if the stack is empty, false otherwise
   */
  boolean isEmpty();

  /**
   * Inserts an element at the top of the stack.
   * @param e the element to be inserted
   */
  void push(E e);

  /**
   * Returns, but does not remove, the element at the top of the stack.
   * @return top element in the stack (or null if empty)
   */
   E top();

  /**
   * Removes and returns the top element from the stack.
   * @return element removed (or null if empty)
   */
   E pop();
}
```

Name (First and Last): Colin Cam D




```
1  class CompStack implements Stack<Integer> {
2    private Node head;
3    private int size;
4
5    public CompStack() {
6      head = null;
7      size = 0;
8    }
9
10   public int size() {
11     return size;
12   }
13
14   public boolean isEmpty() {
15     return this.size == 0;
16   }
17
18   public void push(Integer e) {
19     if (isEmpty()) {
20       head = new Node(e);
21     } else {
22       if (head.data == e) {
23         head.c++;
24       } else {
25         Node nh = new Node(e);
26         nh.next = head;
27         head = nh;
28       }
29     }
30     size++; }
31   }
```

```
32   public Integer top() {
33     if (isEmpty()) return null;
34     else return head.data;
35   }
36
37   public Integer pop() {
38     if (isEmpty()) return null;
39     else {
40       Integer r = head.data;
41       head.c--;
42       if (head.c == 0) {
43         head = head.next;
44       }
45       size--;
46       return r;
47     }
48   }
49
50   private class Node {
51     public Integer data;
52     public int c;
53     public Node next;
54
55     public Node(Integer data) {
56       this.data = data;
57       this.c = 1;
58       this.next = null;
59     }
60   }
61 }
```