Midterm 1 (raw points)

Graded

Student

Colin Cano

Total Points

18 / 33 pts

Question 1

Q1 **3** / 4 pts

1.1 Q1.1 **3** / 3 pts

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

- **– 1 pt** For 1 incorrect or missing number out of 3
- **– 2 pts** For 2 incorrect or missing numbers out of 3
- **– 3 pts** For 3 incorrect or missing numbers out of 3

Q1) 1.Correct

1.2 Q1.2 **0** / 1 pt

- **– 0 pts** The options below are in **alternative** to one another
- **– 0.5 pts** For missing Hat0Scarf0 or Hat1Scarf1 (but not both) or having them in the wrong order
- **– 1 pt** For missing both Hat0Scarf0 and Hat1Scarf1
- **– 0.5 pts** The id variable was not incremented properly.
- **– 0.25 pts** For missing the parentheses around the id.

3.The first Stamper (s) adds "Hat" to both y.label and z.label, assigning incrementing numbers based on id.
The second Stamper (t) adds "Scarf" to both labels, again with incrementing numbers

Question 2

Q2 **6** / 8 pts

2.1 **Q2.1 A or B** **3** / 4 pts

+ 0 pts If the student has answered **both** A and B, pick one arbitrarily and ignore the other.
The options below are **cumulative**.

✔ **+ 0.5 pts** **A.** `x` points to a `Store` instance
B. `g` points to a `Store` instance

+ 0.5 pts **A.** Field `x.emps` points to an array
B. Field `g.emps` points to an array

✔ **+ 0.5 pts** **A.** The array has 4 elements
B. The array has 2 elements

✔ **+ 0.5 pts** **A.** The first two elements of the array are null, the next two point to distinct `Employee` instances.
B. The first element of the array points to an `Employee` instances, the second is null.

✔ **+ 0.5 pts** **A.** Field `boss` of `Employee` instance `x.emps[2]` is null
B. Field `boss` of `Employee` instance `g.emps[0]` is not null

✔ **+ 0.5 pts** **A.** Field `id` of `Employee` instance `x.emps[2]` is 13
B. Field `id` of `Employee` instance `g.emps[0]` is 20

+ 0.5 pts **A.** Field `boss` of `Employee` instance `x.emps[3]` points to the other `Employee` instance: `x.emps[2]`
B. Field `boss` of other `Employee` instance (`g.emps[0].boss`) is null

✔ **+ 0.5 pts** **A.** Field `id` of `Employee` instance `x.emps[3]` is 12
A. Field `id` of `Employee` instance `g.emps[0].boss` is 20

2.2 Q2.2 C or D 3 / 4 pts

+ 0 pts If the student has answered **both** C and D, pick one arbitrarily and ignore the other.
Simulate the given code and grade its generated diagram.
The options below are **cumulative**.

- ✔ **+ 0.5 pts** **C:** code is proper Java (except for minor syntax errors)
- ✔ **+ 0.5 pts** **C:** Code creates 3 instances of `Employee` with respective ids 15, 16 and 17
- **+ 0.5 pts** **C:** `d` points to one of three `Employee` instance with id 15
- **+ 0.5 pts** **C:** `c` has value null.
- ✔ **+ 0.5 pts** **C:** `boss` field of `Employee` instance with id 15 points to `Employee` instance with id 16
- ✔ **+ 0.5 pts** **C:** `boss` field of `Employee` instance with id 16 points to `Employee` instance with id 17
- ✔ **+ 1 pt** **C:** `boss` field of `Employee` instance with id 17 points to `Employee` instance with id 15
- **+ 0.5 pts** **D:** there is one instance of `Store`
- **+ 0.5 pts** **D:** there is one instance of `Employee`
- **+ 0.5 pts** **D:** `a` points to the `Store` instance
- **+ 0.5 pts** **D:** `b` points to the `Employee` instance
- **+ 0.5 pts** **D:** `c` is null
- **+ 0.5 pts** **D:** field `emps` of the `Store` instance is null
- **+ 0.5 pts** **D:** field `boss` of the `Store` instance points to that instance
- **+ 0.5 pts** **D:** field `id` of the `Store` instance is 14

Question 3

Q3

5 / 9 pts

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

✔ **+ 2.5 pts** The answer is non-blank and contains a sensible (but possibly incorrect) Java method, as opposed to nonsense, Python or pseudo code.

+ 0.5 pts Method works correctly (although possibly inefficiently) in the empty list case.

✔ **+ 0.5 pts** Method creates new array to store stretched list.

✔ **+ 0.5 pts** The new array, if any, is double the size of the **list** (not of the old array).

✔ **+ 0.5 pts** Method has a (for or while) loop.

✔ **+ 0.5 pts** The index variable of the loop is initialized correctly.

+ 0.5 pts The loop end condition has the correct upper bound on the chosen array index (note that there are alternative but equally correct implementations).

✔ **+ 0.5 pts** The loop increments the index correctly.

+ 1 pt The loop body correctly sets the even positions of the new array.

+ 1 pt The loop body correctly sets the odd positions of the new array.

+ 0.5 pts The method updates elems with the new array.

+ 0.5 pts The method updates size with the new size.

Question 4

Q4 **3** / 9 pts

4.1 Q4.1 **3** / 5 pts

+ **0 pts** Options below are in **alternative** to one another.

+ **5 pts** Exemplary: identifies the correct problem, and clearly and specifically illustrates what the code actually does.

+ **4 pts** Satisfactory: identifies the correct problem; diagram shown but diagram and/or explanation could be clearer/more specific, or vice versa.

✔ + **3 pts** Progressing: identifies the correct problem; explanation or diagram is given but not both; or could be much clearer than it is; or has errors in exposition.

+ **2 pts** Unsatisfactory but on the right track: Gives some indication of identifying the right problem; explanation not quite there or incomplete.

+ **1 pt** Misses the problem completely.

+ **0 pts** Blank or nonsense.

It is good to notice the special case, however in general cases, it is still incorrect. Also, there is no instance variable tail.

4.2 Q4.2 **0** / 4 pts

✔ + **0 pts** Options below are **cumulative**.

+ **1.25 pts** Says to update second-to-last's next to null.

+ **0 pts** Gives some explanation of how to find the second-to-last Node. (points given in following two rubric items)

+ **1 pt** Mention of leveraging iteration with .next to find second to last node.

+ **0.75 pts** Mention of trait that is needed to find second to last node (something such as node.next.next is null)

+ **0.5 pts** Identifies (or reasonably implies) how the result will still be returned properly. (mention of a process such as using a temporary variable)

+ **0.5 pts** Identifies the special case of list size of 1.

Question 5

Q5 **1** / 3 pts

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

+ **3 pts** Fully correct implementation, with double loop.

+ **2 pts** Minor errors but got the idea of needing two nested loops with base index and offset index.

✔ + **1 pt** substantial errors but uses two nested loops.

+ **0 pts** Incorrect or missing implementation.

Name (First and Last): Colin Cano

CS2230 Computer Science II: Data Structures

Midterm Exam I

March 4, 2025

You may have 1 page 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.

Take a deep breath … and relax.

Grading

Question	Points
1	4
2	8
3	9
4	9
Total	30

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).

Question 0

If you have questions for the exam proctor(s) during the test. Other than that, you are not allowed to receive or give help in the taking of this exam.

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 didnt reciever or give help in taking the exam. Colin Cano

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

Name (First and Last): Colin Cano

1. Object-oriented programming (4 points)

Consider the following Java code.

```
class Factory {
    public Factory() { }

    public void run() {
        Stamper s = new Stamper("Hat");
        Stamper t = new Stamper("Scarf");
        Product y = new Product();
        Product z = new Product();
        s.stamp(y);
        s.stamp(z);
        t.stamp(y);
        t.stamp(z);
        System.out.println(y.label);
        System.out.println(z.label);
    }

    public static void main(String[] args) {
        Factory f = new Factory();
        f.run();
    }
}
```

```
class Stamper {
    private String text;
    private int id;

    public Stamper(String s) {
        this.text = s;
        this.number = 0;
    }

    void stamp(Product p) {
        p.label = p.label + text +
                  "(" + id + ")";
        id += 1;
    }
}

class Product {
    public String label;
    public Product() { label = ""; }
}
```

(3 points) How many of each type of object are created by running `Factory.main`?

Stamper	Product	Factory
11	11	1

(1 point) What will the program print?

11 11
11 11

2. Box-and-arrow diagrams (8 points)

Consider these Java classes.

```
class Employee {
    Employee boss;
    int id;

    public Employee(int id) {
        this.id = id;
    }
}
```

```
class Store {
    Employee[] emps;

    public Store(int n) {
        emps = new Employee[n];
    }
}
```

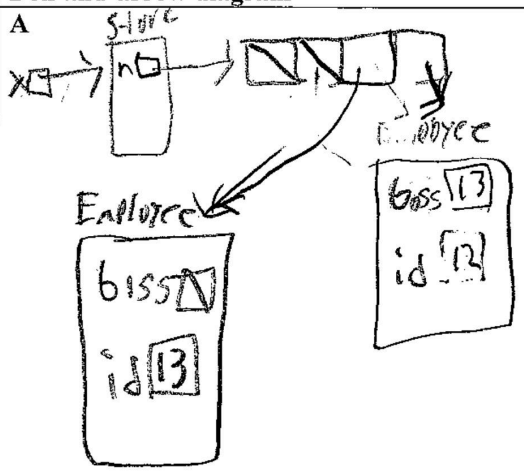
1. (4 points) Pick **one** of the two programs below and draw the box-and-arrow diagram that results from executing the program.

```
class employee
    employee boss;
    int id;
    public Employee(int id){
        this.id = id;

class Store{
    Emp
```

Name (First and Last): Colin Cano

I want this graded (circle one choice) (A) **or** **B**

Program	**Box-and-arrow diagram**

`void main() {`
`  Store x = new Store(4);`
`  x.emps[2] = new Employee(13);`
`  x.emps[3] = new Employee(12);`
`  x.emps[3].boss = x.emps[2];`
`}`
	A
	 

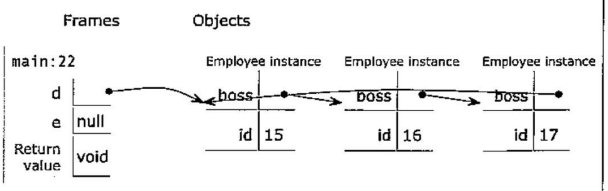
`void main() {`
`  Store g = new Store(2);`
`  g.emps[0] = new Employee(17);`
`  g.emps[0].boss = new Employee(20);`
`  g.emps[0].id = g.emps[0].boss.id;`
`  g.emps[0].boss.boss = g.emps[1];`
`}`
	B

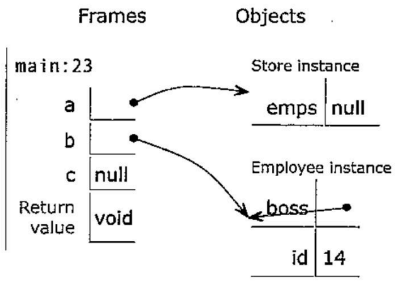
Name (First and Last): Colin Cano

2. (4 points) Pick **one** of the box-and-arrow diagrams below and write a (correct) Java program that would create it.

I want this graded (circle one choice) **C** **or** **D**

Program	**Box-and-arrow diagram**

C
Store main=new Store(3);
main.d = new Employee(15);
main.d.boss=new Employee(16);
main.d.boss[0] =new Employee(17);
main.d.boss.boss[0] = main.d[0];
	

D	

Name (First and Last): Colin Cano

3. Array lists in Java (9 points)

Consider the `ArrayList<E>` class we have studied so far which implements the generic list data type. Recall that it keeps track of the list's size in the field `size` and stores the list elements in order in the array field `elems`.

Provide an implementation for the method `stretchTwo` below which replaces each element in the list with 2 copies of that element, while maintaining the original order.

For example, if a variable `ls` of type `ArrayList<Integer>` stored the list `[5, 0, 1]`, the effect of the call `ls.stretchTwo()` would be to have `ls` store the list `[5, 5, 0, 0, 1, 1]`.

Calling `stretchTwo` on an empty list has no effect. *Your implementation should use only as much memory as necessary each time*. Other than, that *focus on correctness*, not on performance.

Hint: Observe that, for each position i in the original list, the element at that position ends up at positions $2 \cdot i$ and $2 \cdot i + 1$ in the new list.

```
class ArrayList<E> {
    private E[] elems;    int size = 0;

    ArrayList() { elems = (E[]) new Object[8]; /* starts with an 8-cell array */}

    public void stretchTwo () {
        E[] result = new E[2 · elems.length];
        for( int i = 0; i > elems.length; i++){
            result[i] = elems[i];
            result[i+1] = elems[i];   //duplicates element
        }

    }
}
```

Name (First and Last): Colin Cem O

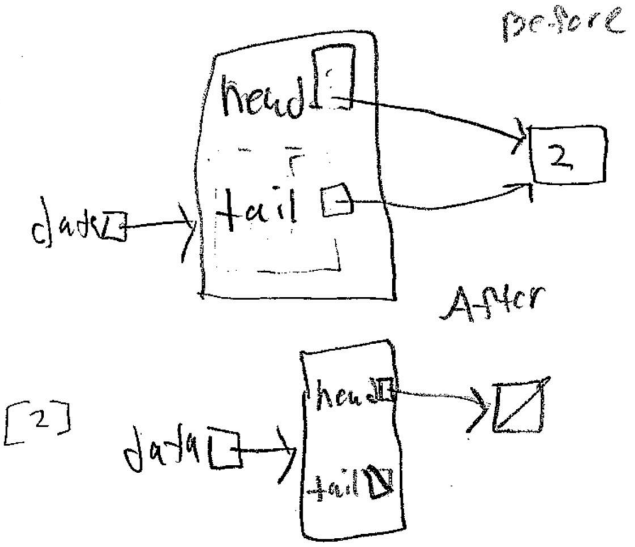
4. Linked lists in Java (9 points)

Consider the generic `LinkedList<E>` class below, which represents a list of elements of any type `E`. The `removeLast` method is expected to remove the last element from the list and return that element. However, the method is incorrect as implemented.

```
class LinkedList<E> {
  private Node head;

  private class Node {
    E data;
    Node next;

    Node(E d) {
      data = d;
      next = null;
    }
  }

  public LinkedList() { head = null; }

  public E isEmpty() { return head == null; }

  private Node findLast() {
    Node n = head;
    while (n.next != null) { n = n.next; }
    return n;
  }

  public void add(E x) {
    if (isEmpty()) {
      head = new Node(x);
    } else {
      Node last = findLast();
      last.next = new Node(x);
    }
  }

  public E removeLast() {
    if (isEmpty()) {
      throw new IllegalStateException("Cannot remove from empty list");
    } else {
      Node last = findLast();
      E result = last.data;
      last = null;
      return result;
    }
  }
}
```

(5 points) What's wrong with `removeLast`? Include an example list and box-and-arrow diagram in the following page to support your answer.

RemoveLast doesn't update head if the list only has 1 node.



Name (First and Last): Colin Cano

(4 points) How would you fix `removeLast`? An explanation in plain English is enough for full credit but it should be specific and sufficiently detailed. If, in alternative, you provide code, be sure to comment on how it fixes the problem.

add an if statement to check if the list is empty and if so, update the head to be null.

5. Optional, extra credit (3 points)

Provide an implementation for the generalization of the method `stretchTwo` from Question 3 that takes as input an `int` *k* and replaces, in order, each element in the list with *k* copies of that element.

```
public void stretch (int k) {
        int[] result=new int[elems.length * K];
        for(int i= 0; i< elems.length; i++){
            int j=0;
            while(i<K){
                result[i]= result[K];
                j++;
            }
        }
}
```