15-121
Fall 2021 Midterm Exam
October 5, 2021

Name:

Andrew ID:

- You may not use any books, notes, or electronic devices during this exam.
- Show your work on the exam to receive credit.
- You may complete the problems in any order you'd like; you may wish to start with the free response problems, which are worth most of the credit.
- All code samples run without crashing unless we state otherwise.
- Assume any imports are already included as required.

Don't write anything in the table below.

Question	Points	Score
1	10	
2	10	
3	16	
4	16	
5	16	
6	16	
7	16	
Total:	100	

1. **Multiple Choice**

For each of the following, choose the best answer.

(a) (2 points) Imagine that you have a class, `Dog` that inherits from another class, `Pet`. In `Dog` you override `toString`. Which of the following lines of code, when executed inside the `toString` of `Dog`, will call the `toString` from `Pet`?

- ○ `super(Dog).toString()`
- ○ `super(Pet).toString()`
- ○ `Dog.toString()`
- ○ `Pet.toString()`
- ○ `super().toString()`
- ○ `super.toString()`

(b) (2 points) Which one of the following is an example of a checked exception?

- ○ `NumberFormatException`
- ○ `NullPointerException`
- ○ `FileNotFoundException`
- ○ `ArithmeticException`
- ○ `ArrayIndexOutOfBoundsException`

(c) (2 points) By convention, your instance variables should usually be...

- ○ `public`
- ○ `protected`
- ○ `private`
- ○ `static`

(d) (2 points) If you want an instance variable to be accessible to both the parent and the child, but not most other classes, then you should declare it...

- ○ `public`
- ○ `protected`
- ○ `private`
- ○ `static`

(e) (2 points) In homework 1, which of the following lines of code adds one item to the `intArray`? (Assuming there is enough space.)

- ○ `intArray[numValues++] = newItem;`
- ○ `intArray.add(newItem);`
- ○ `intArray.append(newItem);`
- ○ `intArray.addHead(newItem);`
- ○ It was impossible to add items to the `intArray` in homework 1.

2. (10 points) **Code Tracing**

Consider the following two classes, and indicate what is printed when the main method of the B class is executed. Place your answer (and nothing else) in the box under the code.

```
public class A {
    protected int a;
    protected int b;

    public A(int a, int b) {
        this.a = b;
        this.b = a;
        System.out.println("A1: " + a + " " + b);
        System.out.println("A2: " + this.a + " " + this.b);
    }

    public int getOne() {
        return this.a;
    }

    public int getTwo() {
        return this.b;
    }
}

public class B extends A {
    public B(int c, int d) {
        super(d, c);
        System.out.println("B1: " + a + " " + b);
        a = d + 6;
        b = c - 6;
        System.out.println("B2: " + this.a + " " + this.b);
    }

    public int getOne() {
        return b;
    }

    public int getTwo() {
        return a;
    }

    public static void main(String[] args) {
        B bob = new B(22, 31);
        int a = bob.getOne();
        int b = bob.getTwo();
        System.out.println(a + " " + b);
    }
}
```

3. (16 points) **Big-Oh**

Consider the following class containing four methods. Determine the big-oh runtime of each of the methods, in terms of N, the number of items in `list`. Write your answer, and nothing else, in the box next to each method.

```java
public class BigOh {
    private ArrayList<String> list;



    public BigOh(ArrayList<String> arg) {
        list = arg;
    }




    /*
     * For the purposes of efficiency analysis, assume that
     * this.list and target have the same number of elements.
     */
    public ArrayList<String> f1(ArrayList<String> target) {
        ArrayList<String> r = new ArrayList<String>();
        for (String item: target) {
            if (this.list.contains(item)) {
                r.add(item);
            }
        }
        return r;
    }



    public int f2(String s) {
        int r = -1;
        for (int i = 0; i < this.list.size(); i++) {
            if (this.list.get(i).equals(s)) {
                r = i;
            }
        }
        return r;
    }




    public void f3(String s) {
        int t = f2(s);
        while (t != -1) {
            this.list.set(t, "121");
            t = f2(s);
        }
    }

}
```

4. **Linked List Memory Diagram**

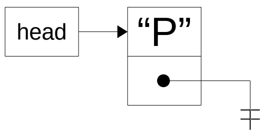
Consider the following program that creates a linked list. You may assume that the `ListNode` class exists and was defined as in class. (If you really don't remember `ListNode`, you can see it defined as part of Problem 7 in this exam.)

```java
public class NodeCT {
    public static void main(String[] args) {
        ListNode<String> head = null;
        ListNode<String> a = new ListNode<String>("M");
        ListNode<String> b = new ListNode<String>("Q");
        ListNode<String> t = null;
        ListNode<String> z = new ListNode<String>("L");

        head = new ListNode<String>("P");
        t = head;
        t.next = a;
        t = t.next;
        t.next = b;
        z.next = new ListNode<String>("M");
        t = new ListNode<String>("V");
        t.next = head;
        head = t;
        a = head.next.next.next;
        b = head.next.next;
        b.next = z;
        z.next.next = a;
    }
}
```

Draw the state of the linked list after the execution of each specified line of code. (The linked list is defined as starting at `head`.) The first one is drawn for you.

(a) After Line 9



(b) (4 points) After Line 11

(c) (4 points) After Line 14

(d) (4 points) After Line 17

(e) (4 points) After Line 21

5. (16 points) **Free Response**: Birds!

Consider the following testcases for two different classes:

```
public static void main(String[] args) {
    boolean ret;

    // A Bird has a name, and initially has no eggs and isn't flying
    Bird b = new Bird("Pigeon");
    System.out.println(b); // Prints: Pigeon [0 eggs] is not flying

    // A bird can lay eggs one at a time
    ret = b.layEgg();
    System.out.println(ret); // Prints: true
    ret = b.layEgg();
    System.out.println(ret); // Prints: true
    System.out.println(b); // Prints: Pigeon [2 eggs] is not flying

    // A bird can also fly
    b.fly();
    System.out.println(b); // Prints: Pigeon [2 eggs] is flying

    // A bird can't lay an egg while flying
    ret = b.layEgg();
    System.out.println(ret); // Prints: false
    System.out.println(b); // Prints: Pigeon [2 eggs] is flying

    // A flying bird can land
    b.land();
    System.out.println(b); // Prints: Pigeon [2 eggs] is not flying

    // A Penguin is a special type of Bird
    Penguin p = new Penguin();
    System.out.println(p instanceof Bird); // Prints: true
    System.out.println(p instanceof Penguin); // Prints: true
    System.out.println(p); // Prints: Penguin [0 eggs] is not flying

    // Penguins can lay eggs
    ret = p.layEgg();
    System.out.println(p); // Prints: Penguin [1 eggs] is not flying

    // But Penguins can't fly...
    p.fly(); // Causes:
    // Exception in thread "main" java.lang.IllegalStateException: Penguins cannot fly
}
```

Write the classes `Bird` and `Penguin`. Do not hardcode against the values used in the testcases, though you can assume the testcases cover the needed functionality. For full credit, you must use proper object-oriented design, including good inheritance and avoiding unnecessary code duplication.

Write your answers on the next two pages.

Write the class `Bird` on this page.

Write the class Penguin on this page.

6. (16 points) **Free Response**: Broken Contacts

Consider a file containing information about contacts in the format `Name,Age,Email`. Here is an example of a file with two people in it:

```
Ryan Riley,39,rdriley@andrew.cmu.edu
Bob Jones,24,bob@andrew.cmu.edu
```

Also consider the following `Contact` class:

```
public class Contact {
    private String name;
    private int age;
    private String email;

    public Contact(String name, int age, String email) {
        this.name = name;
        this.age = age;
        this.email = email;
    }
}
```

Write a `ContactList` class which has the following features:

- A `ContactList` stores `Contacts` in an `ArrayList`. (The `ArrayList` contains `Contact` objects.)
- The constructor for a `ContactList` takes a single argument, a filename. It reads in the contacts from that file and loads them into the `ArrayList`.
- A contact is valid if it contains a name, age (in numeric numbers), and an `@andrew.cmu.edu` email address. If there is an invalid contact in the file, then that contact should not be loaded into the contact list.

For example, consider the following file, `list.txt`:

```
Ryan Riley,39,rdriley@andrew.cmu.edu
Joe Smith,13,joe@gmail.com
Bob Jones,24,bob@andrew.cmu.edu
Ahmed Al-Thani,Sixteen,ahmed@andrew.cmu.edu
```

You can create a new `ContactList` from this file by calling:

```
ContactList myList = new ContactList("list.txt");
```

That list will contain two users:

- Ryan Riley, age 39, rdriley@andrew.cmu.edu
- Bob Jones, age 24, bob@andrew.cmu.edu

Joe will not be included, because his email address was not `@andrew.cmu.edu`. Ahmed will not be included, because his age was not a numeric number.

Write your answer on the next page

Write the `ContactList` class.

Hint: When verifying that contacts are valid, consider writing and using some helper functions.

7. **Free Response**: Linked Lists

   Consider the following partial code for a Linked List:

```java
public class MyLinkedList<ListType> {
    public ListNode<ListType> head = null;

    public MyLinkedList() {
        this.head = null;
    }

    /* Add a node to the head of the list in O(1) time. */
    public void addHead(ListType item) {
        // Code omitted, but assume this method works properly.
    }

    /* Add a node to the end of the list in O(N) time. */
    public void add(ListType item) {
        // Code omitted, but assume this method works properly.
    }

    /**
     * Add all the items from an ArrayList to this linked list, in order.
     * This function should run in O(N) time.
     *
     * @param arr The ArrayList containing the items to add to the list.
     */
    public void addList(ArrayList<ListType> arr) {
        // You will write this method.
    }

    /**
     * Create an ArrayList containing all of the items from this linked list.
     * This function should run in O(N) time.
     *
     * @return An ArrayList containing all of the items from this LinkedList, in
     *         order.
     */
    public ArrayList<ListType> asList() {
        // You will write this method.
    }

}

public class ListNode<NodeType> {
    private NodeType data;
    public ListNode<NodeType> next;

    public ListNode(NodeType data) {
        this.data = data;
    }

    public NodeType getData() {
        return this.data;
    }
}
```

(a) (8 points) Write the code for the method `addList`.

```
public void addList(ArrayList<ListType> arr) {
```

(b) (8 points) Write the code for the method `asList`.

```
public ArrayList<ListType> asList() {
```