15-121
Fall 2019 Midterm Exam
October 8, 2019

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	15	
2	5	
3	15	
4	15	
5	20	
6	30	
7	0	
8	0	
Total:	100	

1. **Short Answer**

   Answer each of the following *in just a few words.*

   (a) (2 points) What are getters and setters and why do we use them?

   (b) (2 points) Briefly describe how to resize an array in Java in order to make it bigger.

   (c) (2 points) Give a specific example (i.e., name the exception) of one checked exception and one unchecked exception in Java.

   (d) (1 point) Assuming that Roudha is a student in this class, is the following an academic integrity violation? Answer "Yes" or "No", then justify your answer.

   Roudha's friend (who is not in the class) helps her with homework 1 by discussing it with her at a high-level, but does not provide her with a solution or look at her code.

(e) (2 points) The standard convention in Java is to declare all of your instance variables private. Why is this a good idea?

(f) (2 points) In the real world, why is it important to write code that follows a code standard?

(g) (4 points) Write **four lines** of code that finish the method below. (Note: Lines containing only a { or } do not count as lines.) If you can't solve it in four lines, then feel free to use more lines for half-credit.

```
// Return an ArrayList containing all the elements of arr squared.
// For example: [1,2,3] becomes an ArrayList containing 1, 4, and 9.
public static ArrayList<Integer> doubleArray(int[] arr) {




}
```

2. (5 points) **Code Tracing**

Indicate what the code will print. Place your answer (and nothing else) in the box below the code.

```
public class CT {
    public String s;

    public CT(String s) {
        String r = "";
        int m = Integer.parseInt(s.substring(0, 1));
        for (int i = m; i < s.length(); i++) {
            r += s.charAt(i);
        }
        for (int i = m - 1; i > 0; i--) {
            r += s.charAt(i);
        }
        this.s = r;
    }

    public String toString() {
        return s;
    }

    public static void main(String[] args) {
        CT myCT = new CT("3pickle");
        System.out.println(myCT);
    }
}
```

3. (15 points) **Big-Oh**

Determine the big-oh runtime of each of the following, in terms of N, the length of the string. Write your answer, and nothing else, in the box next to each function.

```
public static int func1(String s, char c) {
    int a = 0;
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) == c) {
            a++;
        }
    }
    return a;
}
```

```
public static int func2(String s, char c) {
    int a = 0;
    int i = s.indexOf(c, 0);
    while (i != -1) {
        a++;
        i = s.indexOf(c, i + 1);
    }
    return a;
}
```

```
public static int func3(String s) {
    String z = "aeiou";
    int a = 0;
    for (int i = 0; i < z.length(); i++) {
        if (s.indexOf(z.charAt(i)) != -1) {
            a++;
        }
    }
    return a;
}
```

4. **Linked List Memory Diagram**

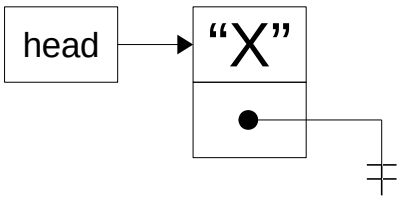
Consider the following program that creates a linked list. You may assume that the `ListNode` class exists and was defined as in class. (If you have forgotten it, you can find a copy in the handout.)

```
1  public class LinkedListDiagram {
2
3      public static void main(String[] args) {
4          ListNode<String> head = null;
5          ListNode<String> a = new ListNode<String>("W");
6          ListNode<String> b = new ListNode<String>("X");
7          ListNode<String> c = new ListNode<String>("Y");
8          ListNode<String> d = null;
9
10         head = b;
11         b.next = a;
12         b.next = null;
13         c.next = b;
14         a.next = c;
15         head = a;
16         d = new ListNode<String>("Z");
17         d.next = head;
18         head = d;
19         b = c;
20         a = b;
21         head.next.next.next.next = head;
22         head.next.next = null;
23         head = a;
24     }
25 }
```

Starting from the head, draw the state of the linked list after the execution each specified line of code. The first one is done for you.

Note: There is an extra copy of this code in the Reference Handout provided to you with this exam. (So that you don't need to keep flipping pages...)

(a) After Line 10



(b) (4 points) After Line 11

(c) (4 points) After Line 15

(d) (4 points) After Line 18

(e) (3 points) After Line 23

5. **Free Response: Library**

(a) (10 points) Consider the following code used to test a class called `LibraryItem`:

```
public static void testLibraryItem() {
    Customer c1 = new Customer("John Smith");
    Customer c2 = new Customer("Ahmed Abdullah");

    // Create a new library item. It was released in 1960
    LibraryItem thing1 = new LibraryItem("One Fish, Two Fish", 1960);

    // Print out the library item
    // The next line prints: "One Fish, Two Fish(1960)"
    System.out.println(thing1);

    // Checkout the item to John. This should return true.
    if (thing1.checkOut(c1) == false) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }

    // Try to check it out to Ahmed. This should return false, because
    // John already checked it out.
    if (thing1.checkOut(c2) == true) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }

    // Get the Customer that has the item checked out.
    // If no one has it checked out, return null.
    // Someone has this checked out, so this doesn't return null.
    Customer tmpCustomer = thing1.checkedOutTo();
    if (tmpCustomer == null) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }
    // The next line prints: "Thing1 is checked out to John Smith"
    System.out.println("Thing1 is checked out to " + tmpCustomer);
}
```

On the next page is a partially completed version of the `LibraryItem` class. Complete it so that it operates as the test code indicates by adding whatever new methods, instance variables, and/or lines of code you require.

```
public class LibraryItem {
    private String title;
    private int year;


    public LibraryItem(String title, int year) {
        this.title = title;
        this.year = year;


    }

    public String toString() {
        return this.title + "(" + this.year + ")";
    }









}
```

(b) (10 points) Now consider the following code used to test a class called `Movie`, which is-a `LibraryItem`:

```
public static void testMovie() {
    Customer c1 = new Customer("John Smith");
    Customer c2 = new Customer("Ahmed Abdullah");

    // Create a new Movie. It was released in 1992 and has a runtime of
    // 123.5 minutes
    LibraryItem thing2 = new Movie("A River Runs Through It", 1992, 123.5);

    // Print out the Movie
    // The next line prints:
    // "Movie: A River Runs Through It(1992)[123.5 minutes]"
    System.out.println(thing2);

    // Checkout the movie to Ahmed. This should return true.
    if (thing2.checkOut(c2) == false) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }

    // Try to check it out to John. This should return false, because
    // John already checked it out.
    if (thing2.checkOut(c1) == true) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }

    // Get the Customer that has the item checked out.
    // If no one has it checked out, return null.
    // Someone has this checked out, so this doesn't return null.
    Customer tmpCustomer = thing2.checkedOutTo();
    if (tmpCustomer == null) {
        System.out.println("Error, this shouldn't happen.");
        return;
    }
    // The next line prints: "Thing2 is checked out to Ahmed Abdullah"
    System.out.println("Thing2 is checked out to " + tmpCustomer);
}
```

On the next page write the `Movie` class so that it operates as the test code indicates. Your solution *must* properly apply the principals of inheritance. Writing a `Movie` class that does not properly inherit from `LibraryItem` and make use of `LibraryItem`'s methods and data will not receive points.

Space for answer to Question 5(b).

6. **Free Response: Lists**

(a) (15 points) Write the static function `listContainsItems` which, given an `ArrayList` of integers and an array of integers, returns `true` if each of the items in the array also appears in the `ArrayList`. It returns `false` otherwise.

For example, if I have an `ArrayList` named `myList` which contains the values `{1,5,19,29,30,45}`, then...

```
int[] arr1 = {45,19,1};
int[] arr2 = {1,19,25};

// The next line returns true because 45, 19, and 1 are each in myList.
listContainsItems(myList, arr1);

// The next line returns false because 25 is not in myList.
listContainsItems(myList, arr2)
```

(b) (15 points) Write the static function `linesInList` which, given a `String` containing a filename and an `ArrayList` containing values, prints out the line number of each line in the file that only contains integers which are also in the `ArrayList`. The file contains lines of integers where each integer is separated by the `'/'` character.

For example, if I have an `ArrayList` named `myList` which contains the values `{1,5,19,29,30,45}`, and a file named `bob.txt` containing the following lines:

```
45/19/1
46/26/48/42
29
35/12/12/38/6
```

then `linesInList("bob.txt", myList)` would print out:

```
1
3
```

Hints:

- You should call `listContainsItems`. You can assume it works even if yours does not.
- You can convert a string to an integer using `Integer.parseInt(String s)`. For example, calling `Integer.parseInt("15")` returns `15`.
- You may assume that you can use the `getFileScanner(String filename)` method given in the notes.

Write your answer on the next page. Do not write it on this page.

Space for Answer to Question 6

7. (1 point (bonus)) **Extra Credit: Code Tracing**

   Indicate what the code will print. Place your answer (and nothing else) in the box below the code.

```
public class LinkedListCT {

    public static void mystery1(ListNode n) {
        if (n == null) {
            return;
        }
        mystery1(n.next);
        System.out.print(" " + n.data + " ");
    }

    public static void mystery2(ListNode n) {
        if (n == null) {
            return;
        }
        System.out.print(" " + n.data + " ");
        mystery2(n.next);
    }

    public static void main(String[] args) {
        ListNode head = null;
        head = new ListNode("A");
        head.next = new ListNode("B");
        head.next.next = new ListNode("C");
        mystery1(head);
        mystery2(head);
    }

}
```

8. (3 points (bonus)) **Extra Credit: Reasoning over Code**

   Determine an appropriate argument to pass to `ROC` that causes it to return `true`. Write your answer (the value for `n`), and nothing else, in the box below the code.

```
public static boolean ROC(int n) {
        String s = "";
        while (n > 0) {
                s += (char) ((n % 10) + 'a');
                n = n / 10;
        }
        s = "4" + s;
        String x = new CT(s).toString(); // See Q2
        return (x.equals("feedbad"));
}
```