

## CIS 110 — Spring 2022 — Exam 2

Full Name: \_\_\_\_\_

Recitation #: \_\_\_\_\_

Penn ID (e.g. 12345678): \_\_\_\_\_

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

Instructions are below. Not complying will lead to a 0% score on the exam.

- Do not open this exam until told by the proctor.
- You will have exactly 110 minutes to take this exam.
- There is a separate Appendix that will be handed out & contains some code that you will need to refer to. Nothing you write here will be graded.
- Make sure your phone is turned OFF (not on vibrate!) before the exam starts.
- Fill out the information and declaration above. Write your Penn ID (the numbers!!!), eg. 12345678, on every page in the space given before you work on anything. If you don't do so, we won't be able to find your answers and you won't get the points you deserve.
- Food, gum, and drink are strictly forbidden. Masks are mandatory. Can drink water through a straw.
- Green PennOpen passes are required. If you have a non-compliance Red Pass, you cannot take the exam and will get a 0 (no makeup).
- You may not use your phone or open your bag for any reason, including to retrieve or put away pens or pencils, until you have left the exam room.
- This exam is closed-book, closed-notes, and closed computational devices.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All code must be written in proper java format, including all curly braces and semicolons.
- Do not separate the exam pages. Do not take any exam pages with you. The entire exam packet must be turned in as is.
- Only answers on the FRONT of pages will be graded. There are two blank pages at the end of the exam if you need extra space for any graded answers. You may use the back of pages for additional scratch work.
- Use a pencil, or blue or black pen to complete the exam.
- If you have any questions, raise your hand and a proctor will come to you.
- When you turn in your exam, you may be required to show your PennCard. If you forgot to bring your ID, talk to an exam proctor immediately.

Q1	Q2	Q3	Q4	Q5
20 pts	24 pts	18 pts	22 pts	26 pts

## Q1. Short & Sweet (10 Questions, 2 pts each)

**All your answers must be placed in the Answer box below each question. Only answers placed in the designated area will be graded.**

### Question 1.1:

True or False? The following test will pass:

```
@Test
public void testArrays() {
    int[] array1 = {1, 2, 3};
    double[] array2 = {1.0, 2.0, 3.0};
    assertEquals(array1, array2);
}
```

Answer:

True       False

### Question 1.2:

True or False? Before throwing an exception, we must first instantiate it using the **new** keyword.

Answer:

True       False

### Question 1.3:

True or False? When a class implements an interface, you will get a runtime error if you do not implement every method listed in the interface.

Answer:

True       False

### Question 1.4:

True or False? The compareTo method of the comparable interface *should* return 0 if the two objects are not equal.

Answer:

True       False

## Question 1.5:

True or False? The following traverses through the entire 2-D array called arr and replaces all values with 1:

```
int[][] arr = {{1, 2, 3}, {4}, {6, 8}};
for (int i = 0; i < arr.length; i++) {
    for (int j = arr[i].length; j >= 0; j++) {
        arr[i][j] = 1;
    }
}
```

Answer:

 True False

## Question 1.6:

What will the array { 34, 8, 14, 51, 32, 21 } look like after the second “insertion” step of Insertion sort?

- A) 8, 14, 51, 32, 21, 34
- B) 8, 34, 14, 51, 32, 21
- C) 8, 14, 51, 21, 32, 34
- D) 8, 14, 34, 51, 32, 21

Answer (Select one - A, B, C, D):

 A B C D

## Question 1.7:

True or False? Given `int[][] arr = new int[4][3]`, the type of `arr[2]` is `int[]`.

Answer:

 True False

***Continued on Next Page***

## Question 1.8:

True or False: This test case passes:

```
@Test
public void test (expected = NullPointerException.class) {
    int[][] arr = null;
    assertEquals(arr.length, 0);
}
```

Answer:

True

False

## Question 1.9:

Convert the decimal integer 39 to binary:

Answer:

## Question 1.10:

What is the least significant bit (LSB) of the decimal integer 9,876,543,214?

Answer:

## Q2. Long Fill In The Blank - Piazza Par-tay! (12 questions, 2 pts each)

Sukya and Hannah - avid Piazza enthusiasts - have decided to represent the CIS 110 Piazza through the power of Java code! As you know, Piazza is a platform where people (like yourself) can post questions! When posting a question, the author must include a question (of course), and they must also include which folder(s) the question pertains to.

In the Piazza class, they wish to implement methods that will enable questions to be posted, deleted, or searched for. They have created the stub files below the Question and Piazza classes, but need your help to fill in the blanks!

Each Question has three fields: a String for the content, a String for the author's name, and an ArrayList for the names of the folders. All fields are private and have getters as needed. There is also a simple constructor.

Each Piazza instance has a LinkedList of Questions as its only field.

You can assume that all necessary import statements are included.

**Please fill in the corresponding code for each blank denoted by a \_\_\_#\_\_\_. You can find the code in the Appendix. You need to fill in the blanks and put your answers in the table on the page immediately after the code. Only the answers in the box will be graded.**

**Write your answers here:**

<b>Blank Number</b>	<b>Your Answer</b>
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

### Q3. Tracing - Elevate A Little Higher (9 questions, 2 pts each)

Harnwell is experimenting with a new kind of Elevator which takes Students to their floors at 3x its current speed. The catch, however, is that the elevator is only able to transport 1 student at a time. Thus, to resolve this issue, it has been decided that the Student who lives on the highest floor (of all the students who want to use the elevator) may wait to use the Elevator – everyone else has to take the stairs. Below we have code to represent this new Elevator as an object, as well as Student objects. Each Student has a floor number and an energy level, which changes depending on whether they take the elevator or the stairs.

**You can find the code in the Appendix. There are 3 classes: Student, Elevator, and Main.**

Fill in the boxes below to indicate what will be printed out at each checkpoint when running Main.java.

Checkpoint 1:

<code>alex.getEnergy()</code>	
<code>lydia.getEnergy()</code>	
<code>amy.getEnergy()</code>	
<code>elevator.getHighestFloor()</code>	

Checkpoint 2:

<code>rachel.getEnergy()</code>	
<code>bart.getEnergy()</code>	
<code>kishen.getEnergy()</code>	
<code>elevator.getHighestFloor()</code>	
<code>elevator.getCurrStudent()</code>	

## Q4. Coding Problem 1 - Encryption Schmiction (22 pts)

Katie and Taylor are avid Marvel fans who love the new MoonKnight show, exclusively on Disney+. Moon Knight is the avatar for the Egyptian Moon God, and has been gifted with the power to decrypt ancient Egyptian hieroglyphics. Upset that they don't have this same power, they've decided to write code that gives them this power - but for English of course!

To communicate with each other, Katie and Taylor will send each other an encrypted String message (all uppercase characters) and a 5 by 5 2D array of all uppercase characters. To decrypt or encrypt a given cipher, they will have to do the following:

- For each character in the cipher string, find the pair of indices where the matrix's value is equal to that character
- Find the transposed indices (switch the row and the column - the row number becomes the column number, and vice versa)
- Append the character located at the transposed indices to the output string
- Note that the alphabet has 26 letters, and the 2D array only has 25 elements. Each character can only appear at most once, but one character in the alphabet will still not be located in this array.
  - If the character in the cipher string is this singular letter of the alphabet that is not located in the array, add that same character to the output string as is.

For example, let's say the array of characters is:

```
{{'J', 'A', 'U', 'C', 'I'},
 {'X', 'V', 'B', 'N', 'L'},
 {'M', 'G', 'O', 'Y', 'H'},
 {'S', 'R', 'Z', 'D', 'T'},
 {'K', 'W', 'E', 'Q', 'P'}}
```

Note that the character 'F' is missing from the array in this case. If the message (the cipher) is "FAMILY", it will encrypt to "FXUKWZ". The table below maps out each character to the relevant indices:

Cipher Char	Indices	Transposed Indices	New Char
F	N/A (not in array)	N/A (not in array)	F (same character)
A	[0][1]	[1][0]	X
M	[2][0]	[0][2]	U
I	[0][4]	[4][0]	K
L	[1][4]	[4][1]	W
Y	[2][3]	[3][2]	Z



Decryption works very similarly. For example, this matrix and the encrypted message “EHWWO” will decrypt to “HELLO”. Note that since ‘O’ is at indices [2][2], the encryption does not change this character.

Katie and Taylor need your help to code the encryption and decryption algorithms that you can assume are being written in a class called MatrixCipher! You may assume that nothing is null.

**Question 4.1 (17 pts):** Implement the encrypt method below.

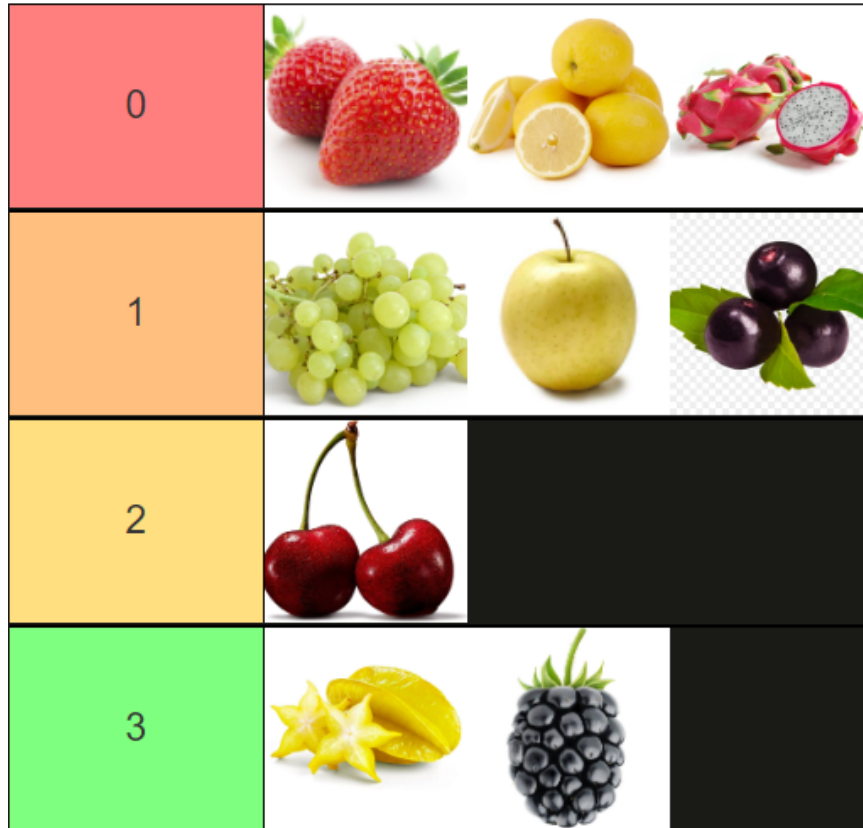
```
/**
 * Iterate through each character in the message. For each
 * character, iterate through the matrix and find that character's
 * position (if it's in the matrix) & append its corresponding
 * transposed character to your output.
 * If you didn't come across the character in the matrix, append it
 * to your output as it is.
 */
public static String encrypt(String message, char[][] matrix) {
    // TODO: implement this
}
}
```

**Question 4.2 (5 pts):** Implement the decrypt method below.

```
public static String decrypt(String cipher, char[][] matrix) {  
    // TODO: implement this in ONE LINE ONLY  
    // If you use more than one line, you will get ZERO points  
  
}
```

## Q5. Coding Problem 2 - Hot Takes, anyone? (26 pts)

Tier lists are fun ways of sharing your preferences with your friends. CIS 110 TAs love to make them! Here's an example tier list of fruits:



A tier list is a sequence of tiers, where the tiers are ordered relative to each other. The first tier in the list (tier 0) contains the set of fruits that are superior to all other fruits, according to the tier list maker. The following tier (tier 1) contains fruits that are all worse than fruits in tier 0, but better than fruits in tier 2 and tier 3. Note that tiers can't be empty, tiers can't have duplicates, and that the ordering within a tier doesn't matter--the author of the above list is indifferent between lemons, strawberries, and dragonfruit.

Help CIS 110 design some objects for representing tier lists. We'll use two classes to do this: `Tier.java` and `TierList.java`. A `Tier` is like a node in a `Tour` from HW7: it stores some data (a `String[]` representing the entries at this Tier) and a pointer to the next Tier. A `TierList` is a linked sequence of `Tier` objects. This is like the `Tour` itself. Read the provided methods to see what methods are available for you to use in your implementation of the missing methods.

The course staff put together most of `TierList.java` and `Tier.java` for you, but we need your help in figuring out if two `TierList` objects are structurally equal to each other.

***Continued on Next Page***



## Question 5.2 (18 pts): TierList equals()

Two TierList objects a and b are equal if they have the same number of tiers, and if each of the tiers at each position in the sequence are equal to each other (using Tier's .equals()). Implement .equals() in TierList.java.

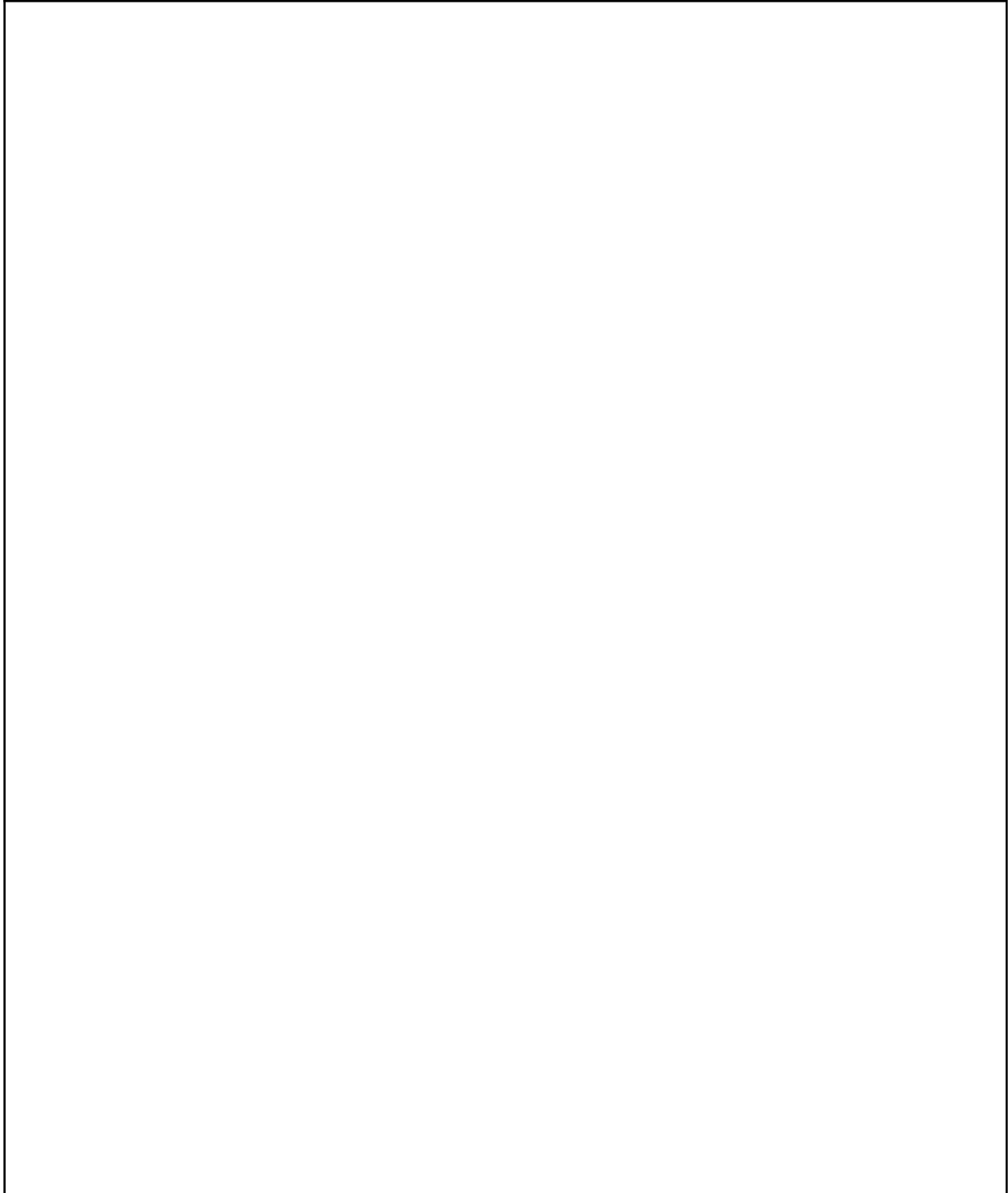
**You can find the code for the TierList class in the Appendix. You have to only implement one method in the space below.**

```
public boolean equals(TierList other) {
```

```
}
```

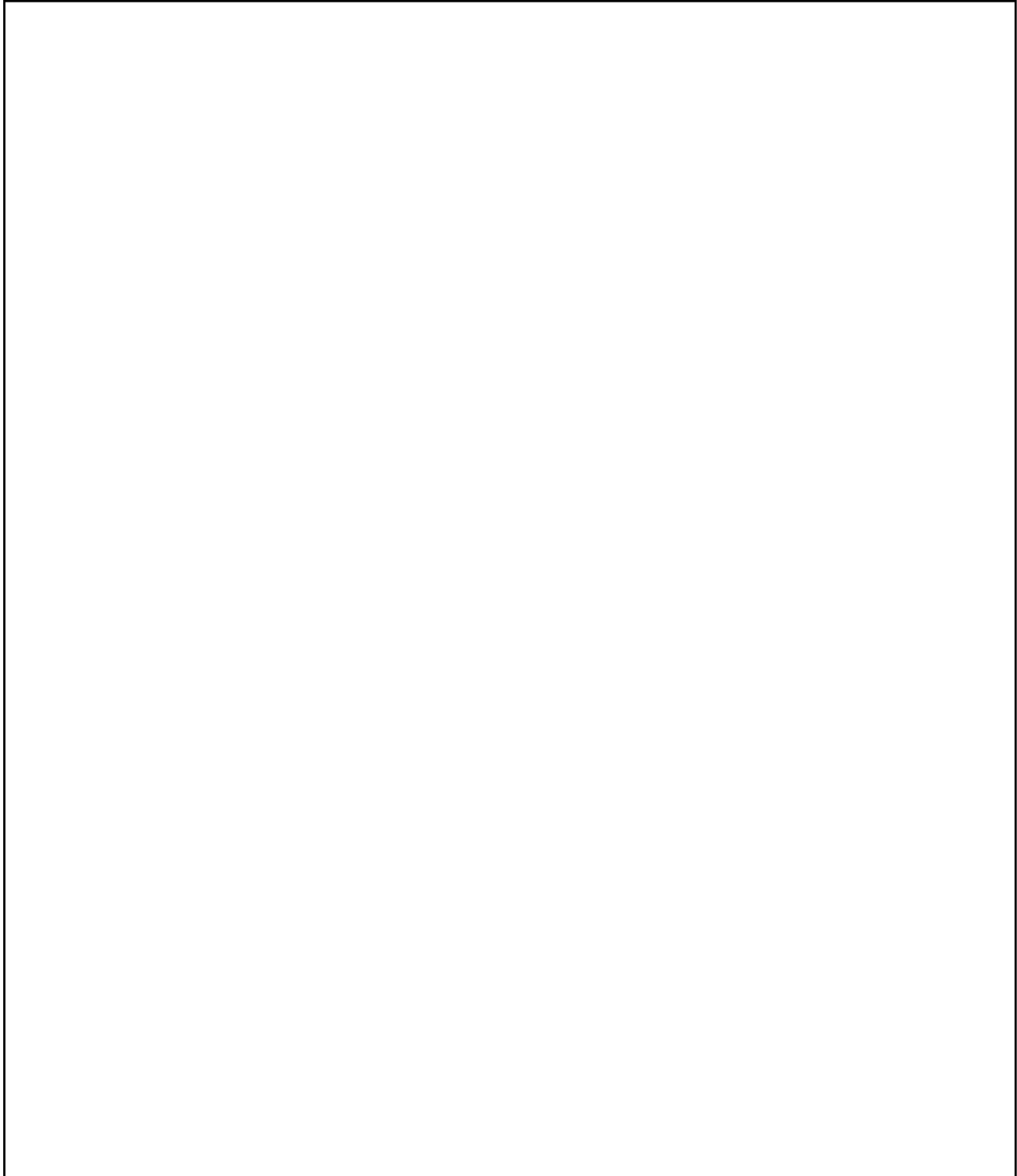
**Extra Answers Page (This page is intentionally blank)**

You may use this page for additional space for answers; keep it attached to this exam. Clearly note on the original question page that your answer is on this extra page, and clearly note on this page what question you are answering.

A large, empty rectangular box with a thin black border, occupying most of the page below the instructions. It is intended for students to write their answers to exam questions.

**Extra Answers Page (This page is intentionally blank)**

You may use this page for additional space for answers; keep it attached to this exam. Clearly note on the original question page that your answer is on this extra page, and clearly note on this page what question you are answering.

A large, empty rectangular box with a thin black border, occupying most of the page below the instructions. It is intended for students to write their answers to exam questions.

CIS 110 — Spring 2022 — Exam 2 APPENDIX

Full Name: \_\_\_\_\_

Recitation #: \_\_\_\_\_

Penn ID (e.g. 12345678): \_\_\_\_\_

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

\_\_\_\_\_  
Signature

\_\_\_\_\_  
Date

**This document is an Appendix and contains code that you will need to refer to to answer questions. Nothing you write here will be graded. You may also use this as scratch paper.**

NOT GRADED



## Q2. Long Fill In The Blank - Piazza Par-tay!

```
public class Question {
    private String questionContent;
    private String author;
    private ArrayList<String> folders;

    // constructor
    _____1_____ (String content, String postAuthor,
                       ArrayList<String> folders) {
        questionContent = content;
        author = postAuthor;
        _____2_____ = folders;
    }

    public ArrayList<String> getFolders() {
        return folders;
    }

    public String getAuthor() {
        return author;
    }
}
```

***Please see the next page for more code for Q2.***

```
public class Piazza {
    private LinkedList<Question> questions;

    public Piazza() {
        // there are no questions at first
        questions = _____3_____;
    }

    public void postQuestion(String author, String questionContent,
        ArrayList<Folder> folders) {
        Question question = new Question(___4___, ___5___, folders);

        // add the question to the list of questions
        _____6_____;
    }

    public void deleteFirstQuestionByAuthor(String author) {
        for (int i = 0; i < questions._____7_____; i++) {
            Question question = _____8_____;

            // check if this question's author is the input author
            if (_____9_____) {
                // remove the question at specified index
                _____10_____;
                break;
            }
        }
    }

    public LinkedList<Question> findQuestionsInFolder(String folderName) {
        LinkedList<Question> output = new LinkedList<>();
        for (Question question : questions) {
            // check if the question is in the input folder
            _____11_____ folders = question.getFolders();
            if (folders._____12_____) {
                output.add(question);
            }
        }
        return output;
    }
}
```

### Q3. Tracing - Elevate A Little Higher

```
public class Student {
    private int energy;
    private int floorNumber;

    public Student(int energy, int floorNumber) {
        if (floorNumber < 1 || floorNumber > 24) {
            throw new IllegalArgumentException("Invalid Floor");
        }
        this.energy = energy;
        this.floorNumber = floorNumber;
    }

    public void setEnergy(int newEnergy) {
        this.energy = newEnergy;
    }

    public int getEnergy() {
        return this.energy;
    }

    public int getFloor() {
        return this.floorNumber;
    }
}
```

***Please see the next page for more code for Q3.***

```
public class Elevator {
    private int currentHighestFloor;
    private Student currStudent;

    public Elevator() {
        this.currentHighestFloor = 0;
        this.currStudent = null;
    }

    // A new student wants to use the elevator
    public void addStudent(Student s) {
        if (currStudent == null) {
            this.currStudent = s;
        } else {
            if (s.getFloor() > this.currentHighestFloor) {
                Student p = this.currStudent;
                this.currStudent = s;
                this.currentHighestFloor = s.getFloor();
                p.setEnergy(p.getEnergy() - p.getFloor());
            } else {
                s.setEnergy(s.getEnergy() - s.getFloor());
            }
        }
    }

    // The elevator has arrived & will take the student with the current
    // highest floor to their floor. The queue will then reset.
    public void takeStudentToFloor() {
        this.currStudent.setEnergy(this.currStudent.getEnergy() + 10);
        this.currStudent = null;
        this.currentHighestFloor = 0;
    }

    public int getHighestFloor() {
        return this.currentHighestFloor;
    }

    public Student getCurrStudent() {
        return currStudent;
    }
}
```

***Please see the next page for more code for Q3.***

```
public class Main {
    public static void main(String[] args) {
        Elevator elevator = new Elevator();

        Student alex = new Student(100, 7);
        Student lydia = new Student(80, 20);
        Student amy = new Student(90, 10);

        elevator.addStudent(alex);
        elevator.addStudent(lydia);
        elevator.addStudent(amy);

        // Checkpoint 1
        System.out.println(alex.getEnergy());
        System.out.println(lydia.getEnergy());
        System.out.println(amy.getEnergy());
        System.out.println(elevator.getHighestFloor());

        Student rachel = new Student(60, 4);
        Student bart = new Student(96, 6);
        Student kishen = new Student(100, 19);

        elevator.takeStudentToFloor();
        elevator.addStudent(rachel);
        elevator.addStudent(bart);
        elevator.addStudent(kishen);
        elevator.takeStudentToFloor();

        // Checkpoint 2
        System.out.println(rachel.getEnergy());
        System.out.println(bart.getEnergy());
        System.out.println(kishen.getEnergy());
        System.out.println(elevator.getHighestFloor());
        System.out.println(elevator.getCurrStudent());

    }
}
```

## Q5. Coding Problem 2 - Hot Takes, anyone?

```
import static java.util.Arrays.*;

public class Tier {
    private String[] entries;
    private Tier next;

    public Tier(String[] entries) {
        this.entries = entries;
        this.next = null;
    }

    public String[] getEntries() {
        return this.entries;
    }

    public void setNext(Tier next) {
        this.next = next;
    }

    public Tier getNext() {
        return this.next;
    }

    public boolean equals(Tier other) {
        // TODO: implement this
    }
}
```

***Please see the next page for more code for Q5.***

```
public class TierList {

    private Tier tierZero;

    /*
     * Do not worry too much about the implementation of this constructor. You
     * don't need to use it.
     */
    public TierList(String[][] allEntries) {
        Tier current = null;
        for (int i = 0; i < allEntries.length; i++) {
            String[] tierArray = allEntries[i];
            if (tierArray.length > 3) {
                throw new IllegalArgumentException("That's cheating!");
            }
            Tier nextTier = new Tier(tierArray);
            if (current == null) {
                tierZero = nextTier;
                current = nextTier;
            } else {
                current.setNext(nextTier);
                current = current.getNext();
            }
        }
    }

    public Tier getTierZero() {
        return this.tierZero;
    }

    public boolean equals(TierList other) {
        // TODO: Implement this
    }
}
```

**Scratch Paper (This page is intentionally blank)**

Nothing you put here will be graded.

NOT GRADED



**Scratch Paper (This page is intentionally blank)**

Nothing you put here will be graded.

NOT GRADED