Solutions to Final

Problem 1: Warm Up Problems

(1.1) What is the difference between pass by value and pass by reference? Why is this important?
- **Passing by value**: Some version of: When a variable is passed by value, a copy is made of the original value and passed into the method.
- **Passing by reference**: Some version of: When a variable is passed by reference, a reference to the value’s memory address is passed into the method.
- **This is important because** (one of, or similar):
  - This changes the way methods interact with different parameters.
  - Objects are passed by reference and primitives are passed by value.
  - We don’t have to return things passed by reference.
  - etc.

(1.2) What is the difference between a parameter and a return value?
- **Parameter**: Some version of: A parameter is a value that is passed into a method.
- **Return Value**: Some version of: A return value is the value that is given back out of the method.

(1.3) True/False:
1. A HashMap is passed by value. **False**
2. An int is passed by value. **True**
3. ArrayLists can store primitives and Objects. **False**
4. You can alter a String using .charAt(index). **False**
5. A class defines a new variable type. **True**

(1.4) Reading Code:
**Part 1**: What is deepFriedOreo equal to?
- 0, 2, 4, 6, 8

**Part 2a**: What is deepFriedOreo equal to?
- 0, 4, 6, 8

**Part 2b**: What is cottonCandy equal to?
- \([0 \ 0 \ 8 \ 0],\)
- \([0 \ 0 \ 0 \ 0],\)
- \([0 \ 0 \ 0 \ 0],\)

**Part 3a**: What is deepFriedOreo equal to?
- 0, 4, 6, 8, 0

**Part 3b**: What is cottonCandy equal to?
- \([0 \ 0 \ 8 \ 0],\)
- \([0 \ 2 \ 2 \ 0 \ 0],\)
- \([0 \ 3 \ 0 \ 0],\)
Problem 2: Raffle Tickets

/*
 * File: Raffle.java
 * ---------------------
 * Solution for Problem 2 on the Summer 2019 CS106A Final.
 */

import acm.program.*;

public class RaffleTickets extends ConsoleProgram {

    public void run() {
        // Test cases
        checkRaffleDistance("ABCDEF", "ABDCEF");
        checkRaffleDistance("1A2B3C4D", "00000000");
        checkRaffleDistance("1", "B");
        checkRaffleDistance("1Q2W3E", "1Q2W3E");
    }

    /**
     * Given two values, will compute the distance and print size of the prize
     * a player wins.
     * @param winningVal
     * @param yourVal
     */
    private void checkRaffleDistance(String winningVal, String yourVal) {
        int distance = 0;
        for(int i = 0; i < winningVal.length(); i++) {
            if(winningVal.charAt(i) != yourVal.charAt(i)) {
                distance += 1;
            }
        }

        println("Your raffle code is a distance of " + distance + " away!");
        if(distance == 0) {
            println("Congratulations, you're the winner!");
        } else if(distance == 1 || distance == 2) {
            println("Congratulations, you win a small prize!");
        } else {
            println("Sorry, you didn't win any prizes.");
        }
    }
}

Problem 3, Part 1: Designing the Ride Class

/*
 * File: Ride.java
 * ---------------------
 * Solution for Problem 3, Part 1 on the Summer 2019 CS106A Final.
 */

import acm.program.*;
import java.util.*;

public class Ride{
    // private instance variables for Ride class
    private String name;
    private String category;
    private ArrayList<String> guestsOnRide;
    private boolean isWorking;

    // Constructor for Ride class
    public Ride(String name, String type) {
        this.name = name;
        this.category = type;
        this.guestsOnRide = new ArrayList<>();
        this.isWorking = true;
    }

    /**
     * Add given guestName to guestsOnRide ArrayList
     * @param guestName
     */
    public void enterRide(String guestName) {
        this.guestsOnRide.add(guestName);
    }

    /**
     * If a guest is on the ride, remove them and return true.
     * Otherwise, return false.
     * @param guestName
     * @return
     */
    public boolean exitRide(String guestName) {
        return this.guestsOnRide.remove(guestName);
    }

    /**
     * Return size of guestsOnRide ArrayList
     * @return
     */
    public int getNumGuests() {
        return this.guestsOnRide.size();
    }

    /**
     * Returns boolean to show if ride is working
     * @return
     */
}
Problem 3, Part 2: Are the Rides Working?

```java
/**
 * Given an array of rides, will return a boolean array which
 * shows which rides are working.
 * @param rides
 * @return
 */
private boolean[] workingRides(Ride[] rides) {
    boolean[] workingRides = new boolean[rides.length];
    for(int i = 0; i < workingRides.length; i++) {
        workingRides[i] = rides[i].getWorkingStatus();
    }
    return workingRides;
}
```
Problem 4: Painting Roller Coasters

/*
 * File: PaintingRollerCoasters.java
 *---------------------
 * Solution for Problem 4 on the Summer 2019 CS106A Final.
 */

import java.awt.Color;
import java.awt.event.*;
import javax.swing.JButton;
import acm.graphics.*;
import acm.program.*;

public class PaintingRollerCoasters extends GraphicsProgram {

    // the filename of the ride image you will use (GIVEN TO STUDENTS)
    private static final String IMAGE_FILE = "ride.png";
    private static final int CANVAS_WIDTH = 800;
    private static final int CANVAS_HEIGHT = 300;
    private GImage ride = new GImage(IMAGE_FILE);

    public void init() {
        add(new JButton("Red"), SOUTH);
        add(new JButton("Green"), SOUTH);
        add(new JButton("Blue"), SOUTH);
        addActionListeners();
    }

    public void run() {
        // resizing/centering was not required
        setCanvasSize(CANVAS_WIDTH, CANVAS_HEIGHT);
        ride.setLocation((CANVAS_WIDTH - ride.getWidth()) / 2.0, (CANVAS_HEIGHT - ride.getHeight()) / 2.0);

        // add image to canvas
        add(ride);
    }

    public void actionPerformed(ActionEvent e) {
        String cmd = e.getActionCommand();
        int targetPixel;
        if (cmd.equals("Red")) {
            targetPixel = GImage.createRGBPixel(255, 0, 0);
        } else if (cmd.equals("Green")) {
            targetPixel = GImage.createRGBPixel(0, 255, 0);
        } else {
            targetPixel = GImage.createRGBPixel(0, 0, 255);
        }

        int[][] pixels = ride.getPixelArray();
        for (int r = 0; r < numRows(pixels); r++) {
            for (int c = 0; c < numCols(pixels); c++) {
                // set any non-white / non-black pixels to target color
                int pixel = pixels[r][c];
                if (!isWhite(pixel) && !isBlack(pixel)) {
                    pixels[r][c] = targetPixel;
                }
            }
        }
    }
}
ride.setPixelArray(pixels);

/**
 * Returns true if a pixel is white, false if not.
 * @param pixel
 * @return
 */
private boolean isWhite(int pixel) {
    int red = GImage.getRed(pixel);
    int blue = GImage.getBlue(pixel);
    int green = GImage.getGreen(pixel);
    return red == 255 && blue == 255 && green == 255;
}

/**
 * Returns true if a pixel is black, false if not.
 * @param pixel
 * @return
 */
private boolean isBlack(int pixel) {
    int red = GImage.getRed(pixel);
    int blue = GImage.getGreen(pixel);
    int green = GImage.getBlue(pixel);
    return red == 0 && blue == 0 && green == 0;
}

private int numRows(int[][] arr) {
    return arr.length;
}

private int numCols(int[][] arr) {
    return arr[0].length;
}
Problem 5: Ice Cream Stand

import acm.program.*;
import java.util.*;

public class IceCreamStand extends ConsoleProgram {
    public void run() {
        HashMap<String, ArrayList<String>> orders = getOrders();
        HashMap<String, Integer> toppingCounts = getToppingCounts(orders);
    }

    private HashMap<String, ArrayList<String>> getOrders(){
        HashMap<String, ArrayList<String>> custToToppings = new HashMap<>();
        println("Welcome to the CS106A Carnival Ice Cream Stand! \n");
        // Get new customers
        String cust = readLine("Customer Name: ");
        while(!cust.equals("") ) {
            // Get toppings for each customer
            ArrayList<String> currToppings = new ArrayList<>();
            String topping = readLine("First Topping: ");
            while((topping.equals("")) ) {
                currToppings.add(topping);
                topping = readLine("Next topping: ");
            }
            custToToppings.put(cust, currToppings);
            cust = readLine("Customer Name: ");
        }
        println("Done getting orders!");
        return custToToppings;
    }

    private HashMap<String, Integer> getToppingCounts(HashMap<String, ArrayList<String>> customerToToppings) {
        HashMap<String, Integer> toppingCounts = new HashMap<>();
        // Loop over customer names
        for(String customer : customerToToppings.keySet()) {
            ArrayList<String> toppings = customerToToppings.get(customer);
            // Loop over topping names and add counts to new map
            for(String topping : toppings) {
                if(toppingCounts.containsKey(topping)) {
                    int newCount = toppingCounts.get(topping) + 1;
                    toppingCounts.put(topping, newCount);
                } else {
                    toppingCounts.put(topping, 1);
                }
            }
        }
        return toppingCounts;
    }
}
Problem 6: Popping Balloons

import java.awt.*;  
import java.util.*;  
import java.awt.event.*;  
import acm.graphics.*;  
import acm.program.*;  
import acm.util.*;  

public class PoppingBalloons extends GraphicsProgram {

    // Below given to students
    private static final String BALLOON_FILENAME = "balloon.png";
    private static final String POPPED_FILENAME = "poppedBalloon.png";
    private static final int Y_OFFSET = 100;
    private static final int BALLOON_SIZE = 75;
    private static final int NUM_BALLOONS = 10;

    private GImage dart = new GImage("dart.png");
    private GLabel poppedBalloonsLabel = new GLabel("Balloons Popped: 0");
    // Above code given to students
    
    private ArrayList<GImage> validBalloons = new ArrayList<>();
    private RandomGenerator rgen = RandomGenerator.getInstance();
    
    public void run() {
        setup();
        drawBalloons();
    }

    private void drawBalloons() {
        for(int balloon = 0; balloon < NUM_BALLOONS; balloon++) {
            int randX = rgen.nextInt(0, getWidth() - BALLOON_SIZE);
            int randY = rgen.nextInt(Y_OFFSET, getHeight() - BALLOON_SIZE);
            GImage newBalloon = new GImage(BALLOON_FILENAME);
            // you did not need setSize in your solution
            newBalloon.setSize(BALLOON_SIZE, BALLOON_SIZE);
            add(newBalloon, randX, randY);
            validBalloons.add(newBalloon);
        }
    }

    private GImage getRandomBalloon() {
        int index = rgen.nextInt(0, validBalloons.size() - 1);
        GImage balloon = validBalloons.get(index);
        validBalloons.remove(index);
        return balloon;
    }

    public void mouseClicked(MouseEvent e) {
        GImage maybeDart = getElementAt(e.getX(), e.getY());
    }
}
if (maybeDart == dart && validBalloons.size() != 0) {

    GImage currBalloon = getRandomBalloon();

    // Strategy 1: Removing and Replacing Image
    double balloonX = currBalloon.getX();
    double balloonY = currBalloon.getY();
    remove(currBalloon);
    GImage newPoppedBalloon = new GImage(POPPED_FILENAME);
    add(newPoppedBalloon, balloonX, balloonY);

    // Strategy 2: Resetting Current Image Filename
    currBalloon.setImage(POPPED_FILENAME);

    // you did not need to setSize in your solution
    newPoppedBalloon.setSize(BALLOON_SIZE, BALLOON_SIZE);

    poppedBalloonsLabel.setText("Balloons Popped: " + (NUM_BALLOONS - validBalloons.size()));
}

// Below code given to students
public void setup() {
    setBackground(new Color(160, 82, 45));
    poppedBalloonsLabel.setFont("Helvetica-24");
    dart.setSize(BALLOON_SIZE, BALLOON_SIZE);
    add(poppedBalloonsLabel, 0, poppedBalloonsLabel.getAscent());
    add(dart, 0, poppedBalloonsLabel.getHeight());
}