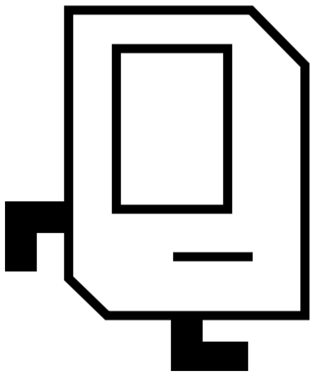
Sarai Gould & Laura Cruz-Albrecht Summer 2019
CS106A July 22nd

CS106A Midterm Exam

This is a closed book, closed note exam. You are, however, allowed 2 pages of notes. You are also allowed a Syntax Reference sheet provided to you at the exam. You will be graded on functionality -- but good style saves time and helps graders understand what you were attempting. You have 180 minutes. We hope this exam is an exciting journey.



I acknowledge and accept the letter and spirit of the honor code. I pledge to write more neatly than I have in my entire life:

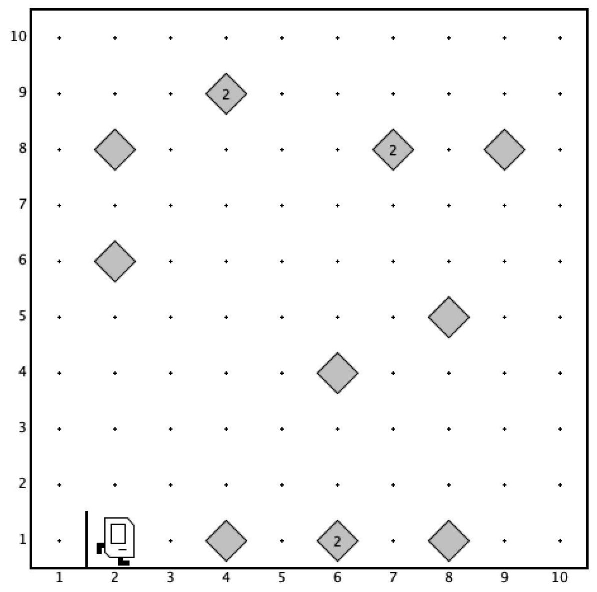
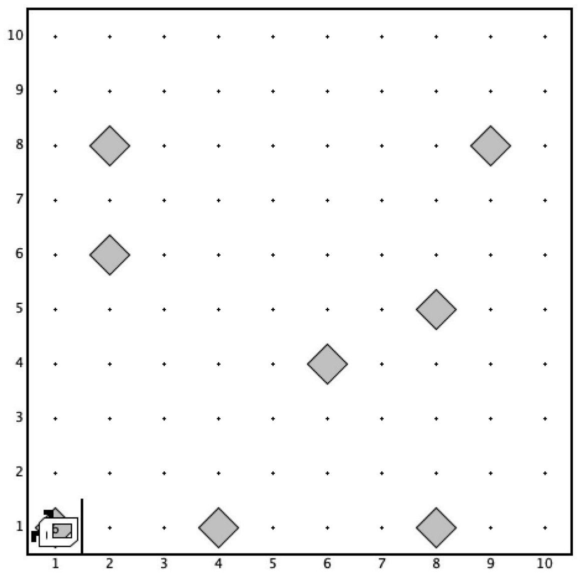
Signature: ______________________________

First Name: ______________________________

Last Name: ______________________________

SUNET: ______________________________

Problem 1: A Day at the Beach [30 points]

 World before Karel has collected shells.	 World after Karel has collected shells. There are 6 shells in the "bucket".

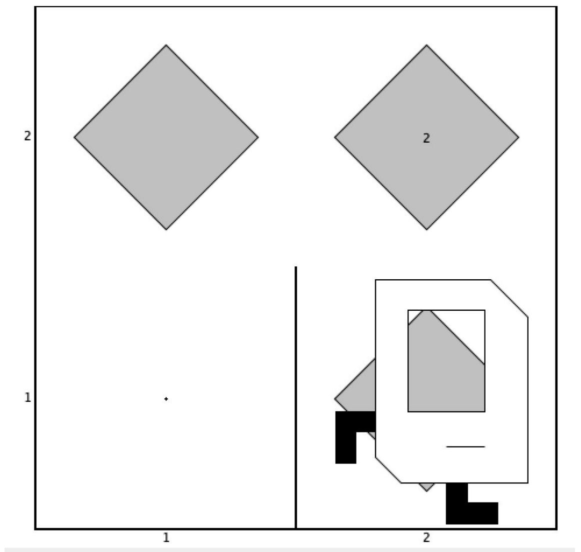
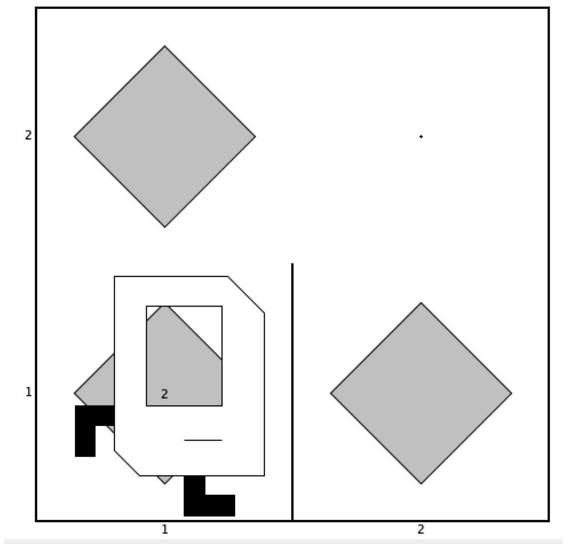
Karel is collecting shells! Out for a day at the beach, Karel is excited to pick up the best shells they can find.

Write a program in which Karel will pick up all of the beepers in piles of height 2, leaving behind beepers in piles of height 1. Karel must then place all of the collected beepers into the "bucket" in space (1,1).

Constraints/Assumptions:

- **You can only use Karel commands on this problem!** No variables, parameters, return statements, breaks, etc.
- Karel will always start at (2,1).
- There will always be a wall between (1,1) and (2,1) creating a "bucket" in the (1, 1) space.
- Karel will always start with an empty beeper bag. There is a Karel command that checks if Karel's bag is empty.
- All piles of "shells" (beepers) will be of either heights 1 or 2. Karel will not see beeper piles taller than height 2.
- Karel's ending location/orientation doesn't matter.
- Your program should work for a world of any width or height, except you can assume it will have width and height of at least 2.

In particular, your program should be able to work for the following world:

 Smallest beach Karel would deign to visit.	 Small beach after Karel has collected shells.

WRITE ANSWER ON NEXT PAGE

```
/* Answer for Problem 1 */

// You can assume all necessary libraries are imported for you.
import stanford.karel.*;

public class DayAtTheBeach extends SuperKarel {
```

```
// Additional space for Problem 1
```

Problem 2: Java Expressions & Tracing [15 points]

This question has two parts.

Part A: For each of the following expressions, indicate their value. Be sure to list the appropriate type (e.g. 7.0 rather than 7 for a double, Strings in double quotes, chars in single quotes true/false for a boolean, etc.):

1. `5 + 7 * 3 / 2 + (1 - 1) * 2` ____________________
2. `‘a’ + “pp” + (double)1 + ‘e’` ____________________
3. `(!true || 9 % 2 > 0) && (44 / 10 == 4.4)` ____________________

Part B: What are the values of `char a`, `int num`, `GOval circle`, and `String dukesPwd` at the end of this program? (The end of the program is notated by the `println("all done!").`)

```
public void run(){
        char a = 'a';
        int num = 5;
        GOval circle = new GOval(100, 100);
        String dukesPwd = "iloveCS106A";
        a = bark(num);
        squeak(circle, dukesPwd);
        println("all done!");
}

private char bark(int num){
        GOval circle = new GOval(150, 150);
        num += 25;
        return 'b';
}

private void squeak(GOval oval, String word){
        oval.setColor(Color.BLUE);
        word.toUpperCase();
}
```

1. What is `a` equal to?

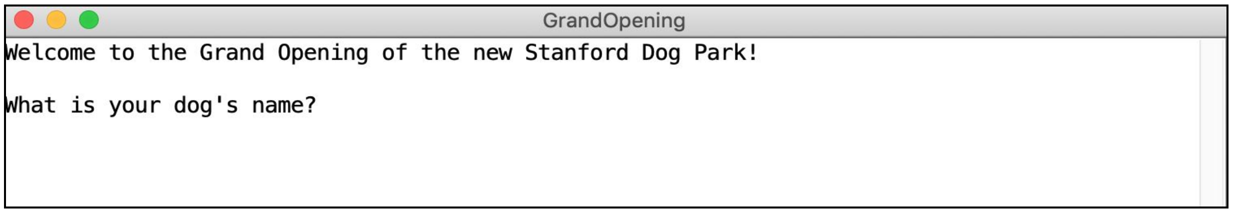
2. What is `num` equal to?

3. What is the width, height, and color of `circle`?

4. What is `dukesPwd` equal to?

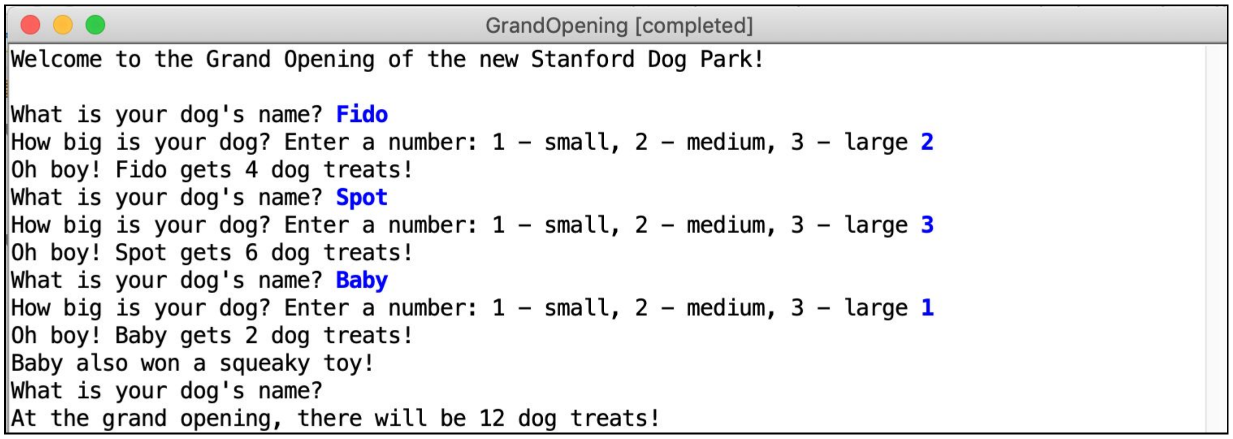
Problem 3: Grand Dog Park Opening [25 points]

Before Receiving User Input:



```
Welcome to the Grand Opening of the new Stanford Dog Park!

What is your dog's name?
```

After Receiving User Input:



```
Welcome to the Grand Opening of the new Stanford Dog Park!

What is your dog's name? Fido
How big is your dog? Enter a number: 1 - small, 2 - medium, 3 - large 2
Oh boy! Fido gets 4 dog treats!
What is your dog's name? Spot
How big is your dog? Enter a number: 1 - small, 2 - medium, 3 - large 3
Oh boy! Spot gets 6 dog treats!
What is your dog's name? Baby
How big is your dog? Enter a number: 1 - small, 2 - medium, 3 - large 1
Oh boy! Baby gets 2 dog treats!
Baby also won a squeaky toy!
What is your dog's name?
At the grand opening, there will be 12 dog treats!
```

Thank you for helping us get ready for the Grand Opening of the new Stanford Dog Park! In order to celebrate this Grand Opening, we're going to buy dog treats for every user who registers their dog. In addition to treats, each dog has a 25% chance of winning a squeaky toy!

Write a Console Program that will help us register these dogs and calculate how many treats we need to buy for the Grand Opening

In Your Program:

- Ask for each dogs' name.
- Ask for the size of each dog in integer form:
  - 1 for small dogs
  - 2 for medium-sized dogs
  - 3 for large dogs
- Dogs will receive the following number of treats:
  - 2 treats for small dogs
  - 4 treats for medium-sized dogs
  - 6 treats for large dogs
- There is a 25% chance that a dog will win a squeaky toy! Calculate for each dog whether or not they won a squeaky toy!

- Tell each user how many treats their dog will receive. You must include the **dog's name** and the **number of treats** in this message. If the dog received a squeaky toy, also include that in your message!
- If the user enters an empty string ("") instead of a dog's name, the program will stop asking for new dogs.
- At the end of the program, it must print the total number of dog treats we will need to buy for the Grand Opening!

Constraints/Assumptions:

- An unlimited number of dogs can register for the Grand Opening. The program will only stop prompting for dogs once the user types in an empty string.
- The 25% chance of winning a squeaky toy must be determined randomly.
- You can assume that the user will enter a valid dog size. The user will never enter a number other than 1, 2, or 3 for their dog's size.

WRITE ANSWER ON NEXT PAGE

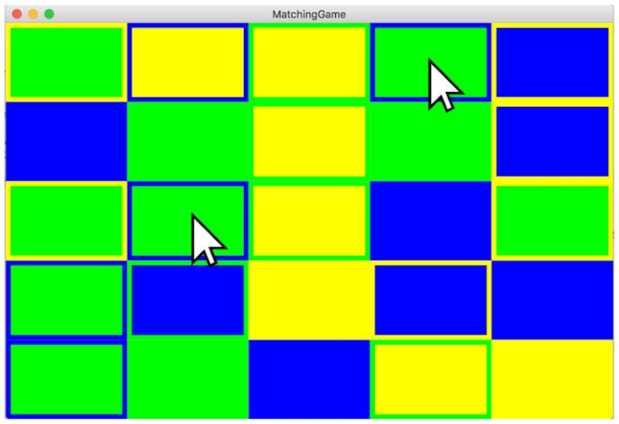
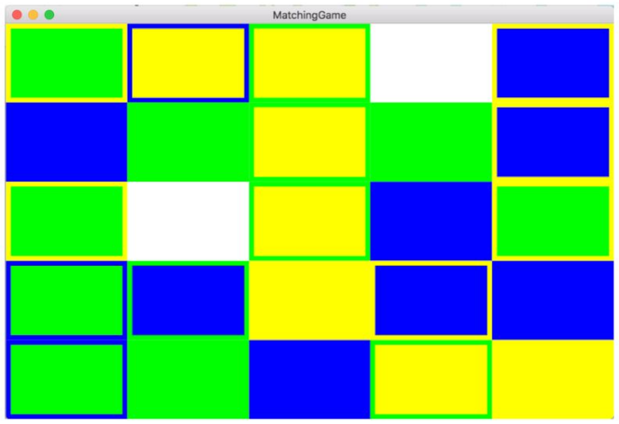
```
/* Answer for Problem 3 */

// You can assume all necessary libraries are imported for you.
import acm.program.*;
import acm.util.*;

public class GrandOpening extends ConsoleProgram {
```

```
// Additional space for Problem 3
```

Problem 4: Match the Colors [30 points]

Most of us have played a sort of matching game as a child (or adult). In this game, you will need to create a grid of colored rectangles, and the user will attempt to select two matching rectangles by clicking on them. If they match, both are then removed. **To make this game easier than the original version, you won't have to "flip" anything. You can already see all of the colors you will need to match with a pair.**

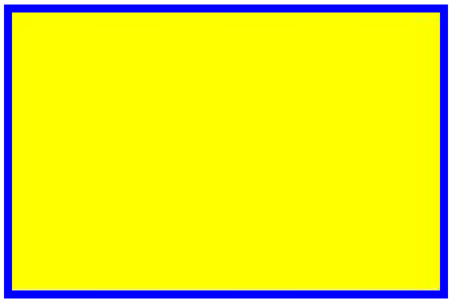
Two boxes are declared as "matching" if both their inner color and outer color match. In the below picture, we match two rectangles with a **green inner color** and a **blue outer color**.

Starting Grid:	**Grid After Matching Two Boxes:**
	

Write a program which implements a matching game where two rectangles match if both their outer and inner color match.

In Your Program:

- Define a constant NUM_ROWS and a constant NUM_COLUMNS. In our example above, NUM_ROW and NUM_COLUMNS are both 5. You must use these constants when drawing your grid of rectangles.
- Scale the rectangles so that your grid fills the whole screen.
- Each rectangle will be made up of two randomized colors. The ***color*** of the rectangle will be a random color and the ***fill color*** will be a random color. By using rectName.setColor(colorName) you can set the outer color of the rectangle. By using rectName.setFillColor(colorName) you can set the inner color of the rectangle. The rectangle to the right has a color of blue and a fillColor of yellow.



- Your rectangles will have random colors and random fill colors. We have given you a method called `getRandomColor()` that will return a random color. You can use it like: `Color colorName = getRandomColor();`
- A user selects a rectangle by clicking on it.
- When a user clicks on a second rectangle, if the color and fill color match on the first rectangle, you will completely remove both rectangles. If the color and fill color do not match on both rectangles, you will clear the user selections, so they can select two new rectangles.

<u>Constraints/Assumptions:</u>

- A user cannot try to match a sequence of more than two rectangles. Clicking on three matching rectangles in a row will not remove all three rectangles, just the first two.
- Your matching game should fill the whole screen. This means your columns and rows should stretch so that they evenly fill the screen. You do not need to worry about the user resizing the window once you have drawn the rectangles.
- You don't need to "flip" anything in this matching game. All colors and rectangles are visible. Once rectangles have been matched, they are then completely removed from the screen.
- You cannot assume that there will be a specific number of colors. You program must work for any number of colors.

WRITE ANSWER ON NEXT PAGE

```
/* Answer for Problem 4 */

// You can assume all necessary libraries are imported for you.
import java.awt.*;
import acm.graphics.*;
import acm.program.*;
import acm.util.*;

// We have given you a method called getRandomColor() that
// will return a random color.
// You can use it like: Color colorName = getRandomColor();

public class MatchingGame extends GraphicsProgram {
```

```
// Additional space for Problem 4
```

Problem 5: Front Coding - CS106A Style [20 points]

Front Coding, or Incremental Encoding, is an algorithm for compressing large amounts of text data! If we have lots of words with the same prefix, front coding can be used to compress this data and represent it in a smaller way.

Example of Prefixes and Suffixes:

Full Word	**Common Prefix** (Beginning of Word)	**Suffix** (End of Word)
"superficial"	"superf"	"icial"
"superfluous"	"superf"	"luos"

Below is an example of how we can use front coding to compress this data:
"6superf*5icial4luos"

Whoa! That's a bit to take in. That being said, this is shorter than storing "superficial superfluous"! As you get more words and even longer prefixes, the more space you end up saving. Yay, more room for cat pictures! Let's look at how we came up with this crazy string!

The CS106 version of this algorithm is as follows:
length of prefix + prefix + "" + length of 1st suffix + 1st suffix + length of 2nd suffix + 2nd suffix*

You will implement a method that will do this for two words! It will return the front coded string. Here are some examples of words your method will have to work for:

Sample Method Call:	**Return Value:**
frontCoding("automate", "automaton");	"7automat*1e2on"
frontCoding("are", "area");	"3are*01a"
frontCoding("cat", "dog");	"0*3cat3dog"

Write a method that will do this front coding process for two words. You are going to write a method that takes in two Strings, implements front coding on them, and returns the front coded String!

You will be implementing the method:

```java
private String frontCoding(String str1, String str2) {
        // TODO: Fill this in

}
```

Constraints/Assumptions:

- str1 always comes before str2 alphabetically. In front coded, the words are sorted alphabetically, and then put through the algorithm. You can assume we have sorted them for you.
- str1 and str2 will each always be strings of at least length 1. They will never be equal to an empty string.
- str1 and str2 will not necessarily share a prefix. This means that the prefix could be the empty string.
- str1 and str2 will never be the same word.
- str1 could be the common prefix. In other words, it's possible that the prefix that str1 and str2 have in common is str1 in its entirety.

WRITE ANSWER ON NEXT PAGE

```
/* Answer for Problem 5 */

// You do not need to implement any other parts of the program
// except for the following method.

private String frontCoding(String str1, String str2){
```