Karel Prepares for the Olympics! (25 points)

Inspired by the events of the 2016 Rio Olympics (where Stanford athletes earned more medals than all but nine countries) as well as the 2018 Men's World Cup, Karel has decided to train for the 2020 Olympics in Tokyo. But in order to become the world's best striker, Karel first needs to learn ball control!

Write a program called StrikerKarel to help Karel learn the basics of the role:

```java
public class StrikerKarel extends SuperKarel
```

Karel starts off at the corner of (1,1), standing on top of the ball (a beeper). Karel has no beepers in their bag and there are no other beepers in the world. There is one "goal" in the final column of the world, represented by an opening in a wall between the last and second-to-last columns. **Karel needs to move the ball from (1,1) to this "goal".**

Two challenges make Karel's task more difficult:

- In order to simulate dribbling, Karel cannot move more than once without the ball touching the ground. That is, the ball must be put down each time Karel changes location.
- In order to simulate defenders, there are walls blocking Karel's potential paths. Karel must move around these walls.

![Diagram showing Karel starting at (1,1) and goal at (10,10)](image)

Assumptions and Specifications

- The world will have at least two columns but may be any height.
- There will be exactly one "goal" in the final column. The goal corner is the only corner in the world that is blocked to the north, east, and south simultaneously.
- At every reachable corner other than the goal, Karel will be able to find an opening in the wall that allows them to move east without first moving west.
- Karel's ending direction does not matter, so long as Karel is in the goal and on top of a beeper.
- **You are limited to the Java instructions shown in the Karel coursera**. This means the only variables allowed are loop control variables used within the control section of a `for` loop. You are not allowed to use syntax like local variables, instance variables, parameters, return values, Strings, `return` or `break`, etc.
For example, the world on the right is **not valid** because if Karel moves north following the red dashed arrow, they will be trapped by walls to the north, south, and east in the column indicated by the red dashed box. Karel would have to move west in order to move east again. Your program **does not** need to handle this kind of world.
You Prepare to Be a Section Leader! (20 points)

Part A: Getting grilled (10 points)

It's time for section, and your students are grilling you on how to evaluate complicated expressions!
For each expression below, determine its final value. Be sure to write a literal value of 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).

```
i. (double)(16 / 5) * 10
ii. 'D' - 'A' == '3'
iii. 2 + 5 + "H" + 2 * 5 + 2
iv. 200 + 19 % 10 - (42 / 10.0 / 2) * 100
v. !(false || 3 != 4) && !(7 / 2.0 < 3.5)
```

Part B: LaLR Lunacy (10 points)

Now you are helping someone in the LaLR, but they have not named their variables or methods descriptively :-(
Trace through the program's execution to figure out what is happening. For each println call along the way, indicate what the program would output at that point in the execution.

**IMPORTANT:** You must enter your answers in the order the program prints them. To this end, each println has a comment indicating which answer it corresponds to. Correct answers in the wrong order will not earn full credit.

```
public void run() {
    int alexandria = 14;
    int ben = 10;
    alexandria = tinky(ben, alexandria);
    println(ben); // ii
    GOval cory = new GOval(19, 87, 19, 91);
    int deray = 66;
    winky(cory, deray);
    println(cory.getHeight()); // iii
    println(deray); // iv
    String elizabeth = "NSP";
    elizabeth.toLowerCase();
    println(elizabeth); // v
}

private int tinky(int alexandria, int ben) {
    alexandria /= 2;
    ben /= 2;
    println(alexandria); // i
    return ben;
```
private void winky(GOval cory, int deray) {
    deray = 76;
    cory.setSize(19, 92);
    deray++;
}
Colin Prepares for the Saturday Crossword! (40 points)

(The series of events described in this problem is 100% true.)

A few weeks ago, Colin got a subscription to the New York Times daily crossword puzzle. He has been having lots of fun solving the easier puzzles but also wants to work up to the harder ones. After one particularly frustrating attempt at a Saturday puzzle, Colin found a series of blog posts documenting one person’s journey to solving a Saturday puzzle independently. In the posts, the blogger describes a Crossword Trainer program he wrote to help him improve.

Colin was intrigued, but he was too busy writing exam questions to write his own Crossword Trainer program. Instead, he wants you to write it for him!

Part A: Test the Teacher (15 points)

Let’s begin by writing the core strategy for the Crossword Trainer program. The program will display a clue to Colin, and he will have to guess the answer. To simulate solving a clue in a partially-completed puzzle, you should randomly reveal some, none, or all of the answer’s letters to Colin before asking for his guess. Write a method called testOneClue:

```java
private boolean testOneClue(String clue, String answer, double revealChance)
```

This method takes three parameters:
- `clue`: a clue to show Colin
- `answer`: the clue’s answer
- `revealChance`: a number between 0 and 1 (inclusive) that serves as the probability that each letter in `answer` will be revealed as part of a hint before Colin makes a guess

This method should return `true` if the Colin’s guess is correct and `false` if not.

Assumptions and Specifications

- The method should begin by printing the crossword clue.
- After displaying the clue, you should print a hint in which each letter of `answer` is revealed with probability `revealChance`. If a letter should not be revealed, you should represent it as a dash "-".
  
  For example, a hint for the answer "PYRE" could be "----E".
- After displaying the hint, you should prompt Colin to enter a guess with "Guess: ".
- Your guess verification should be case-insensitive; that is, capitalization should not affect Colin’s correctness.
- You may assume that all of the letters in `answer` will be capitalized.

```
Probability of revealing a letter? 2
Enter a valid probability between 0 and 1: 0.4
Smelly cheese
--R--
Guess: brie
Correct!
```

```
Viking’s farewell
----E
Guess: FiRe
The answer was: PYRE
```

```
Smartest farm animal
P--
Guess: PIG
Correct!
```

Percent correct: 66.666666666667%
Part B: Feed in the File (25 points)

Now, let's build the rest of our Crossword Trainer. For this part, assume you have a perfect implementation of `testOneClue` from Part A. Here's how it works: You will be provided a text file named "crossword.txt", which has the following format:

```plaintext
[number of clues]
[clue]#[ANSWER]
[clue]#[ANSWER]
[clue]#[ANSWER]
...
```

For example, "crossword.txt" might look like this:

```plaintext
3
Smelly cheese#BRIE
Viking's farewell#PYRE
Smarterst farm animal#PIG
```

You should begin by asking Colin what probability he wants for revealing each letter. Then, test Colin on each clue in "crossword.txt". After Colin has gone through all clues, print the percentage of clues he guessed correctly. You should be able to match the full sample run above.

Assumptions and Specifications

- There will always be at least one clue in the file.
- You should make sure that Colin enters a probability between 0 and 1, inclusive. If not, print "Enter a valid probability between 0 and 1: " and make him re-enter a probability.
- You may assume that Colin will enter a number for the probability.
- You may assume that the '#' character will not appear within any answers.
- The '#' character may, however, appear within clues.
- You should print "Correct!" on correct guesses and "The answer was: [answer]" on incorrect guesses.
- There should be an empty line between clues.
Candidates Prepare for the Midterm Elections! (35 points)

Since winning their respective primaries, candidates for the contested Senate seat in Wyoming have been hard at work campaigning. Moreover, their campaign teams have been hard at work canvassing to gauge public opinion.

You are independent pollster employed by one of the campaign teams to gather data on how their candidate is doing. With midterm elections approaching, you are about to put together your final report... only to find that your intern was actually working for the other candidate and just deleted all of your data!

In a panic to provide something to the campaign manager, you decide to violate the real-life researcher honor code and fabricate all of your data. Sad times :-(

Write a program called PollsterPanic, described below, to generate your fake polling data:

```java
public class PollsterPanic extends GraphicsProgram
```

Part A: Arbitrary Assignments (20 points)

First, generate a grid consisting of 15 equally-sized rectangular zones within the conveniently rectangular state of Wyoming. When you are done creating the grid, you should display the message, "Map created!". This message must be at the top of the screen, centered in the space between the right edge of the map and the right edge of the window.

You should set each zone to a color randomly chosen from the options of red and blue. If it is helpful, you may assume for this problem that these are the only two colors that exist. It is not important whether the zones have their colors filled in on the grid, but it is important that each zone be assigned a color.

As an example, a map that you create might look like this (note that we used a lighter fill color so that you can more clearly see the color assigned to each zone, but you can use any fill color you want, or no fill color):
Part B: Disseminate Deception (15 points)

Now, let's give the campaign manager a way to interpret the results you just made up. When they click on a zone on the map, your program should replace the "Map created!" message with one of three messages:

- "Democrat zone" if a blue zone is selected
- "Republican zone" if a red zone is selected
- "No zone selected" if they clicked somewhere off the grid
*** CS 106A MIDTERM SYNTAX REFERENCE ***

This document lists some of the common methods and syntax that you will use on the exam.

### Karel the Robot (Karel reader Ch. 1-6)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public class Name extends SuperKarel {...}</td>
<td></td>
</tr>
<tr>
<td>turnLeft(); turnRight(); turnAround();</td>
<td>rotates Karel 90° counter-clockwise, clockwise, or 180°</td>
</tr>
<tr>
<td>move();</td>
<td>moves Karel forward in current direction by one square</td>
</tr>
<tr>
<td>pickBeeper();</td>
<td>picks up a beeper if present on Karel's corner; else error</td>
</tr>
<tr>
<td>putBeeper();</td>
<td>places a beeper, if present in beeper bag; else error</td>
</tr>
<tr>
<td>frontIsClear(), frontIsBlocked()</td>
<td>Is there a wall in front of Karel?</td>
</tr>
<tr>
<td>leftIsClear(), leftIsBlocked()</td>
<td>Is there a wall to Karel's left (counter-clockwise)?</td>
</tr>
<tr>
<td>rightIsClear(), rightIsBlocked()</td>
<td>Is there a wall to Karel's right (clockwise)?</td>
</tr>
<tr>
<td>beepersPresent(), noBeepersPresent()</td>
<td>Are there any beepers on Karel's current corner?</td>
</tr>
<tr>
<td>beepersInBag(), noBeepersInBag()</td>
<td>Are there any beepers in Karel's beeper bag?</td>
</tr>
<tr>
<td>facingNorth(), notFacingNorth(), facingEast(), notFacingEast(), facingSouth(), notFacingSouth(), facingWest(), notFacingWest()</td>
<td>Is Karel facing north, south, east, or west?</td>
</tr>
</tbody>
</table>

### RandomGenerator (A&S 6.1)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandomGenerator rg = RandomGenerator.getInstance();</td>
<td></td>
</tr>
<tr>
<td>rg.nextBoolean()</td>
<td>returns a random true/false result;</td>
</tr>
<tr>
<td>rg.nextBoolean(probability)</td>
<td>pass an optional probability from 0.0 - 1.0, or default to 0.5</td>
</tr>
<tr>
<td>rg.nextColor()</td>
<td>a randomly chosen Color object</td>
</tr>
<tr>
<td>rg.nextDouble(min, max)</td>
<td>returns a random real number between min and max, inclusive</td>
</tr>
<tr>
<td>rg.nextInt(min, max)</td>
<td>returns a random integer between min and max, inclusive</td>
</tr>
</tbody>
</table>

### String (A&S Ch. 8)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>String s = &quot;hello&quot;;</td>
<td></td>
</tr>
<tr>
<td>s.charAt(i)</td>
<td>the character in this String at a given index</td>
</tr>
<tr>
<td>s.contains(str)</td>
<td>true if this String contains the other's characters inside it</td>
</tr>
<tr>
<td>s.endsWith(str)</td>
<td>true if this String ends with the other's characters</td>
</tr>
<tr>
<td>s.equals(str)</td>
<td>true if this String is the same as str</td>
</tr>
<tr>
<td>s.equalsIgnoreCase(str)</td>
<td>true if this String is the same as str, ignoring capitalization</td>
</tr>
<tr>
<td>s.indexOf(str)</td>
<td>first index in this String where given String begins (-1 if not found)</td>
</tr>
<tr>
<td>s.lastIndexOf(str)</td>
<td>last index in this String where given String begins (-1 if not found)</td>
</tr>
<tr>
<td>s.length()</td>
<td>number of characters in this String</td>
</tr>
<tr>
<td>s.replace(s1, s2)</td>
<td>a new string with all occurrences of s1 changed to s2</td>
</tr>
<tr>
<td>s.startsWith(str)</td>
<td>true if this String begins with the other's characters</td>
</tr>
<tr>
<td>s.substring(i, j)</td>
<td>characters in this String from index i to j (inclusive)</td>
</tr>
<tr>
<td>s.substring(i)</td>
<td>characters in this String from index i (inclusive) to the end of the String</td>
</tr>
<tr>
<td>s.toLowerCase()</td>
<td>a new String with all lower-case or uppercase letters</td>
</tr>
<tr>
<td>s.toUpperCase()</td>
<td></td>
</tr>
</tbody>
</table>

### Character/char (A&S Ch. 8)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>char c = Character.toUpperCase(s.charAt(i));</td>
<td></td>
</tr>
<tr>
<td>Character.isDigit(ch), .isLetter(ch), .isLowerCase(ch), .isUpperCase(ch), .isWhitespace(ch)</td>
<td>methods that accept a char and return boolean values of</td>
</tr>
<tr>
<td>Character.toString(ch)</td>
<td>accepts a character and returns lower/upper-case version of it</td>
</tr>
</tbody>
</table>

### Integer/Int (A&S Ch. 8)

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>int num = Integer.parseInt(&quot;106&quot;);</td>
<td>accepts a numerical String and returns the value as an int</td>
</tr>
<tr>
<td>Integer.parseInt(String)</td>
<td></td>
</tr>
</tbody>
</table>
Scanner
Scanner input = new Scanner(new File("filename")); // scan an input file
Scanner tokens = new Scanner(string); // scan a string
sc.next(), sc.nextLine() read/return the next token (word) or entire line of input as a string
sc.nextInt(), sc.nextDouble() read/return the next token of input as an int or double
sc.hasNext(), sc.hasNextLine(), ask about whether a next token/line exists, or what type it is,
sc.hasNextInt(), sc.hasNextDouble() without reading it
sc.useDelimiter(String) set the character(s) on which the scanner breaks input into tokens
sc.close() closes the scanner

ConsoleProgram
public class Name extends ConsoleProgram { ... }
readInt("prompt"), Prompts/re-prompts for a valid int or double, and returns it
readDouble("prompt")
readLine("prompt"); Prompts/re-prompts for a valid String, and returns it
readBoolean("prompt", "yesString", "noString"); Returns true if they enter yesString, false if they enter noString
promptUserForFile("prompt", "directory"); Prompts for a filename, re-prompts until input is a file that exists in the
printIn("text"); Prints the given text to the console, followed by a newline (\n').
print("text"); Prints the given text to the console.

GraphicsProgram
public class Name extends GraphicsProgram { ... }
add(shape), add(shape, x, y) displays the given graphical shape/object in the window (at x, y)
getExtent(x, y) returns graphical object at the given x/y position, if any (else null)
setHeight(), getWidth() the height and width of the graphical window, in pixels
pause(ms) halts for the given # of milliseconds
remove(shape); removes the graphical shape/object from window so it will not be seen
setBackgroundColor(color); sets canvas background color

Graphical Objects (A&S Ch. 9)
GRect rect = new GRect(10, 20, 50, 70);
new GImage("filename", x, y) image from the given file, drawn at (x, y)
new GLabel("text", x, y) text with bottom-left at (x, y)
new GLine(x1, y1, x2, y2) line between points (x1, y1), (x2, y2)
new GOval(x, y, w, h) largest oval that fits in a box of size w * h with top-left at (x, y)
new GRect(x, y, w, h) rectangle of size w * h with top-left at (x, y)
obj getColor(), obj.getFillColor() returns the color used to color the shape outline or interior
obj.getX(), obj.getY(), returns the left x, top y coordinates, width, and height of the shape
obj.getWidth(), obj.getHeight() night filled (boolean); whether to fill the shape with color
obj.setFillColor(color); what color to fill the shape with
obj.setColor(color); what color to outline the shape with
obj.setLocation(x, y); change the object's x/y position
obj.setSize(w, h); change the object's width and height
label.setSize(String); changes the text that a GLabel displays
label.getAscent(), label.getDescent() returns a GLabel's ascent or descent from the baseline

Colors
rect.setColor(Color.BLUE);
Color.BLACK, BLUE, CYAN, GRAY, GREEN, MAGENTA, ORANGE, PINK, RED, WHITE, YELLOW
Color name = new Color(r, g, b); // red, green, blue from 0-255

Mouse Events (A&S Ch. 10)
public void eventMethodName(MouseEvent event) { ... } 
events: mouseMoved, mouseDragged, mousePressed, mouseReleased, mouseClicked, mouseEntered, mouseExited
e.getX(), e.getY() the x or y-coordinate of mouse cursor in the window