

Solutions to Practice Midterm

Portions of this handout by Eric Roberts and Patrick Young

Problem 1: Karel the Robot (20 points)

```
/* File: InnerBorderKarel.java */
import stanford.karel.*;

public class InnerBorderKarel extends SuperKarel {

    public void run() {
        moveUpRow();
        for(int i = 0; i < 4; i++) {
            handleBorder();
            nextPosition();
        }
    }

    // Assumes Karel starts one avenue before the first beeper to
    // be placed in this line of the border. Places beepers until
    // Karel reaches a wall, but does not place a beeper on the last
    // corner (where Karel is facing the wall).
    private void handleBorder() {
        move();
        while (frontIsClear()) {
            // We check for any existing beepers, so we don't put
            // two beepers on any of the "corners" of the border
            if (noBeepersPresent()) {
                putBeeper();
            }
            move();
        }
    }

    // Moves Karel up one row while keeping the same orientation
    private void moveUpRow() {
        turnLeft();
        move();
        turnRight();
    }

    // Assumes Karel is facing a wall at the end of line of placed
    // beepers and repositions Karel to be facing in direction of next
    // line in the border of beepers that needs to be placed
    private void nextPosition() {
        turnRight();
        move();
        turnRight();
        move();
        turnRight();
    }
}
```

Problem 2: Simple Java expressions, statements, and methods (20 points)

(2a)	<code>5.0 / 4 - 4 / 5</code>	<hr/>	1.25
	<code>7 < 9 - 5 && 3 % 0 == 3</code>	<hr/>	false
	<code>"B" + 8 + 4</code>	<hr/>	"B84"

(2b) Answer:

```
The 1st number is: 78
The 2nd number is: 73
```

Problem 3: Simple Java programs (25 points)

```
/*
 * File: SecondLargest.java
 * -----
 * This program finds the largest and second largest number
 * in a list entered by the user.
 */

import acm.program.*;

public class SecondLargest extends ConsoleProgram {

    /* Defines the sentinel used to signal the end of the input */
    private static final int SENTINEL = 0;

    public void run() {
        println("This program finds the two largest integers in a");
        println("list. Enter values, one per line, using a "
            + SENTINEL + " to");
        println("signal the end of the list.");

        int largest = -1;
        int secondLargest = -1;
        while (true) {
            int input = readInt(" ? ");
            if (input == SENTINEL) break;
            if (input > largest) {
                secondLargest = largest;
                largest = input;
            } else if (input > secondLargest) {
                secondLargest = input;
            }
        }

        println("The largest value is " + largest);
        println("The second largest is " + secondLargest);
    }
}
```

Problem 4: Using the graphics and random number libraries (35 points)

```
/*
 * File: SimpleFrogger.java
 * -----
 * This program solves the Frogger problem from the practice midterm.
 */

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

/*
 * This program gets a frog to jump one square in the closest
 * direction to a mouse click.
 */
public class SimpleFrogger extends GraphicsProgram {

    public void run() {
        frog = new GImage("frog.gif");
        fx = (NCOLUMNS / 2 + 0.5) * SQUARE_SIZE;
        fy = (NROWS - 0.5) * SQUARE_SIZE;
        add(frog, fx - frog.getWidth() / 2,
            fy - frog.getHeight() / 2);
        addMouseListeners();
    }

    /* Responds to a mouse click */
    public void mouseClicked(MouseEvent e) {
        double mx = e.getX();
        double my = e.getY();
        if (Math.abs(mx - fx) > Math.abs(my - fy)) {
            if (mx > fx) {
                moveFrog(SQUARE_SIZE, 0);
            } else {
                moveFrog(-SQUARE_SIZE, 0);
            }
        } else {
            if (my > fy) {
                moveFrog(0, SQUARE_SIZE);
            } else {
                moveFrog(0, -SQUARE_SIZE);
            }
        }
    }

    /* Moves the frog by dx/dy as long as it remains inside the world */
    private void moveFrog(double dx, double dy) {
        if (insideFroggerWorld(fx + dx, fy + dy)) {
            fx += dx;
            fy += dy;
            frog.move(dx, dy);
        }
    }
}
```

```

/* Returns true if the point (x, y) is inside the frog's world */
private boolean insideFroggerWorld(double x, double y) {
    return (x >= 0 && x <= NCOLUMNS * SQUARE_SIZE &&
           y >= 0 && y <= NROWS * SQUARE_SIZE);
}

/* Private constants */
private static final int SQUARE_SIZE = 75;
private static final int NROWS = 4;
private static final int NCOLUMNS = 7;

/* Private instance variables */
private GImage frog;    /* The image of the frog */
private double fx;     /* The x-coordinate of the frog's center */
private double fy;     /* The y-coordinate of the frog's center */

/* Sets the graphics window size */
public static final int APPLICATION_WIDTH = NCOLUMNS *
SQUARE_SIZE;
public static final int APPLICATION_HEIGHT = NROWS * SQUARE_SIZE;
}

```

Problem 5: Strings and characters (20 points)

```

/*
 * Removes any doubled letters from a string.
 */
private String removeDoubledLetters(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if (i == 0 || ch != str.charAt(i - 1)) {
            result += ch;
        }
    }
    return result;
}

```