1.

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.

private void handleBorder() {

move();

while (frontIsClear()) {

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

private void nextPosition() {

turnRight();

move();

turnRight();

move();

turnRight();

}

}

2. a. 1.25

b. False

c. “B84”

2. The 1st number is: 78

The 2nd number is: 73

3.

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);

}

}

4.

import acm.graphics.\*;

import acm.program.\*;

import java.awt.\*;

import java.awt.event.\*;

private static final int SQUARE\_SIZE = 75;

private static final int NROWS = 4;

private static final int NCOLUMNS = 7;

private double fx;

private double fy;

public static final int APPLICATION\_WIDTH = NCOLUMNS \* SQUARE\_SIZE;

public static final int APPLICATION\_HEIGHT = NROWS \* SQUARE\_SIZE; }

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();

}

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); }

}

}

private void moveFrog(double dx, double dy) {

if (insideFroggerWorld(fx + dx, fy + dy)) {

fx += dx;

fy += dy;

frog.move(dx, dy);

}

}

private boolean insideFroggerWorld(double x, double y) {

return (x >= 0 && x <= NCOLUMNS \* SQUARE\_SIZE && y >= 0 && y <= NROWS \* SQUARE\_SIZE);

}

}

5.

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;

}