Problem 1: Karel the Robot (10 points)

```java
/*
 * File: BreakoutKarel.java
 * -------------------------
 * The BreakoutKarel class solves the problem from the midterm exam.
 */
import stanford.karel.*;
public class BreakoutKarel extends SuperKarel {
    public void run() {
        while (beepersInBag()) {
            if (beepersPresent()) {
                pickBeeper();
                bounce();
            }
            while (frontIsBlocked()) {
                bounce();
            }
            stepDiagonally();
        }
    }
}
/*
 * Causes Karel to perform a ricochet bounce, which requires
 * no more than turning left.
 */
private void bounce() {
    turnLeft();
}
/*
 * Step diagonally. The precondition for this call is that
 * Karel's front must be clear. The postcondition has Karel
 * facing in the same direction.
 */
private void stepDiagonally() {
    move();
    if (leftIsClear() && noBeepersPresent()) {
        turnLeft();
        move();
        turnRight();
    }
}
```
Problem 2: Simple C expressions, statements, and functions (10 points)

(2a) \( \frac{5.0}{4} - \frac{4}{5} \) \hspace{2cm} 1.25

\( 7 < 9 \) \hspace{2cm} false

\( \text{"B"} + 8 + 4 \) \hspace{2cm} \text{"B84"}

(2b) "cabbage"

(2c) To care is human!

Problem 3: Simple Java programs (15 points)

```java
/*
* File: SecondLargest.java
* -----------------------------
* This program finds the largest and second largest values in a list.
*/

import acm.program.*;

public class SecondLargest extends ConsoleProgram {
    public void run() {
        println("This program finds the two largest integers in a");
        println("list. Use " + SENTINEL + " to signal the end of the input.");
        int count = 0;
        int largest = 0;
        int secondLargest = 0;
        while (true) {
            int number = readInt(" ? ");
            if (number == SENTINEL) break;
            count++;
            if (count == 1) {
                largest = number;
            } else {
                if (number > largest) {
                    secondLargest = largest;
                    largest = number;
                } else if (count == 2 || number > secondLargest) {
                    secondLargest = number;
                }
            }
        }
        if (count == 0) {
            println("No values were entered");
        } else {
            println("The largest value is " + largest);
            if (count > 1) {
                println("The second largest value is " + secondLargest);
            }
        }
    }
}

/* Sentinel value to signal end of input */
private static final int SENTINEL = 0;
```
Problem 4: Using the graphics and random number libraries (15 points)

```java
/*
 * File: RandomlyMovingRedCross.java
 * ------------------------------
 * This program solves the practice midterm problem.
 */

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

public class RandomlyMovingRedCross extends GraphicsProgram {

    /* Sets up the program at the beginning */
    public void init() {
        cross = new RedCross();
        add(cross, getWidth() / 2, getHeight() / 2);
        chooseRandomDirection();
        addMouseListeners();
    }

    /* Runs the simulation */
    public void run() {
        while (true) {
            cross.movePolar(VELOCITY, direction);
            pause(PAUSE_TIME);
        }
    }

    /* Called when the mouse is clicked */
    public void mouseClicked(MouseEvent e) {
        if (cross.contains(e.getX(), e.getY())) {
            chooseRandomDirection();
        }
    }

    /* Resets the direction to a random value */
    private void chooseRandomDirection() {
        direction = rgen.nextDouble(0, 360);
    }

    /* Private constants */
    private static final double PAUSE_TIME = 20;
    private static final double VELOCITY = 3;

    /* Private instance variables */
    private RedCross cross;
    private double direction;
    private RandomGenerator rgen = RandomGenerator.getInstance();
}
```
Problem 5: Using the String class (10 points)

```java
/**
 * 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;
}
```