Question 1

(a) Part

```
public static int numberOfLeapYears(int year1, int year2) {
    int count = 0;

    for(int i = year1; i <= year2; i++) {
        if(isLeapYear(i)) {
            count++;
        }
    }

    return count;
}
```

(b) Part

```
public static int dayOfWeek(int month, int day, int year) {
    return (firstDayOfYear(year) + dayOfYear(month, day, year) - 1) % 7;
}
```

Question 2

```
public class StepTracker {
    private int minSteps;
    private int totalSteps;

    private int totalDaysNum;
    private int activeDaysNum;

    public StepTracker(int num) {
        minSteps = num;
        totalSteps = 0;
        totalDaysNum = 0;
        activeDaysNum = 0;
    }

    public void addDailySteps(int num) {
        totalSteps = totalSteps + num;
        totalDaysNum++;

        if (num >= minSteps) {
            activeDaysNum++;
        }
    }

    public int activeDays() {
        return activeDaysNum;
    }

    public double averageSteps() {
        if (totalDaysNum != 0) {
            return (double) totalSteps / totalDaysNum;
        }  else {
            return 0.0;
        }
    }
```

```
}
```

Question 3

(a) Part

```
public ArrayList<String> getDelimitersList(String[] tokens) {
    ArrayList<String> returnList = new ArrayList<String>();
    for (String token : tokens) {
        if (token.equals(openDel) || token.equals(closeDel)) {
            returnList.add(token);
        }
    }

    return returnList;
}
```

(b) Part

```
public boolean isBalanced(ArrayList<String> delimiters) {
    int closedDelCount = 0;
    int openDelCount = 0;

    for(String item : delimiters) {
        if (item.equals(openDel)) {
            openDelCount++;
        }

        if (item.equals(closedDel)) {
            closedDelCount++;
        }

        if (closedDelCount > openDelCount) {
            return false;
        }
    }

    if (closedDelCount == closedDelCount) {
        return true;
    } else {
        return false;
    }
}
```

Question 4

(a) Part

```
public LightBoard(int numRows, int numCols) {
    lights = new boolean[numRows][numCols];

    for (int r = 0; r < numRows; r++) {
        for (int c = 0; c < numCols; c++) {
            double rnd = Math.random();
            lights[r][c] = rnd < 0.4;
        }
    }
```

```
}
```

(b) Part

```
public boolean evaluateLight(int row, int col) {
    int numOn = 0;

    for (int r = 0; r < lights.length; r++) {
        if (lights[r][col]) {
            numOn++;
        }
    }

    if (lights[row][col] && numOn % 2 == 0) {
        return false;
    }

    if (!lights[row][col] && numOn % 3 == 0) {
        return true;
    }

    return lights[row][col];
}
```