1.

(a)

```
public int findFreeBlock(int period, int duration) {
    int blockLength = 0;
    for (int minute = 0; minute < 60; minute++) {
        if (isMinuteFree(period, minute)) {
            blockLength++;
            if (blockLength == duration) {
                return minute - blockLength + 1;
            }
        } else {
            blockLength = 0;
        }
    }
    return -1;
}
```

(b)

```
public boolean makeAppointment(int startPeriod, int endPeriod, int
duration) {
    for (int period = startPeriod; period <= endPeriod; period++) {
        int minute = findFreeBlock(period, duration);
```

```
        if (minute != -1) {
            reserveBlock(period, minute, duration);
            return true;
        }
    }
    return false;
}
```

2.

```
public class Sign {
    private String message;
    private int width;

    public Sign(String m, int w) {
        message = m;
        width = w;
    }

    public int numberOfLines() {
        int len = message.length();
        if (len % width == 0) {
```

```
            return len / width;
        } else {
            return len / width + 1;
        }
    }

    public String getLines() {
        int linesNeeded = numberOfLines();
        if (linesNeeded == 0) {
            return null;
        }
        String signLines = "";
        for (int i = 1; i < linesNeeded; i++) {
            signLines += message.substring((i - 1) * width, i * width) +
";";
        }
        return signLines + message.substring((linesNeeded - 1) * width);
    }
}
```

3.

(a)
```
public void cleanData(double lower, double upper) {
    for (int i = temperatures.size() - 1; i >= 0; i--) {
        double temp = temperatures.get(i);
        if (temp < lower || temp > upper) {
            temperatures.remove(i);
        }
    }
}
```

(b)
```
public int longestHeatWave(double threshold) {
    int waveLength = 0;
    int maxWaveLength = 0;
    for (double temp : temperatures) {
        if (temp > threshold) {
            waveLength++;
            if (waveLength > maxWaveLength) {
                maxWaveLength = waveLength;
            }
        } else {
            waveLength = 0;
        }
    }
```

```
    return maxWaveLength;
}
```

4.

```
public boolean moveCandyToFirstRow(int col) {
    if (box[0][col] != null) {
        return true;
    }
    for (int row = 1; row < box.length; row++) {
        if (box[row][col] != null) {
            box[0][col] = box[row][col];
            box[row][col] = null;
            return true;
        }
    }
    return false;
}
```

```
public Candy removeNextByFlavor(String flavor) {
    for (int row = box.length - 1; row >= 0; row--) {
        for (int col = 0; col < box[0].length; col++) {
            if (box[row][col] != null &&
box[row][col].getFlavor().equals(flavor)) {
                Candy taken = box[row][col];
                box[row][col] = null;
                return taken;
            }
        }
    }
    return null;
}
```