1. Bubble sort’s inner loop scans the unsorted part of the array comparing adjacent items. If it finds adjacent items out of order, then it exchanges them. This causes the largest item to “bubble” up to the “top” of the unsorted part of the array.

At the start of the second iteration (after the first iteration) of the outer loop, the sorted part of the array contains one element:

a) The inner loop scans the unsorted part by comparing adjacent items and exchanging them if out of order.

```
Unsorted Part       Sorted Part

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>20</td>
<td>35</td>
<td>40</td>
<td>60</td>
<td>10</td>
<td>50</td>
<td>45</td>
<td>90</td>
</tr>
</tbody>
</table>

lastUnsortedIndex = 7
```

b) Write the code for the inner, for-loop.
c) Below is the textbook’s code for Bubble sort. What is “wrong” with this code?

```c
void sortArray(int array[], int size) {
    bool swap;
    int temp;
    do {
        swap = false;
        for (int count = 0; count < (size - 1); count++) {
            if (array[count] > array[count + 1]) {
                temp = array[count];
                array[count] = array[count + 1];
                array[count + 1] = temp;
                swap = true;
            }
        }
    } while (swap);
}
```

d) Correct the above Bubble sort code.