Distributed Algorithms and Lower Bounds – Exercise 2

Question 1:

Here is the pseudocode for Lamport's Bakery Algorithm which we saw in the lecture:

```
array flag[N];
array number[N];
initially flag[i] = 0 and number[i] = 0 for all i

code for process i:
lock()
{
1:   flag[i] = 1;
     // Find the maximum number:
2:   max_num = 0;
3:   for j = 1,…,N:
4:        if(max_num < number[j]) then
5:             max_num = number[j];
     // Wait until lower numbers are done:
6:   for j = 1,…,N:
7:        wait until flag[j] == 0 or
                              (number[j],j) < (number[i], i);
     // Enter the critical section
}
unlock()
{
     flag[i] = 0;
}
```

Prove that this algorithm satisfies

(1) Mutual exclusion, and
(2) Deadlock-freedom.

Question 2:

Now let's look at a version of the algorithm that doesn't use the flag array:

```
lock()
{
     // Find the maximum number:
1:   max_num = 0;
2:   for j = 1,…,N:
3:        if(max_num < number[j]) then
4:             max_num = number[j];
     // Wait until lower numbers are done:
5:   for j = 1,…,N:
6:        wait until (number[j],j) < (number[i], i);
     // Enter the critical section
}

unlock()
{
}
```

Show an execution where this modified version fails to guarantee mutual exclusion.