Prepare your answers to the following questions either in a plain text file or in a Microsoft Word file. Answer each question clearly and concisely, but completely, using complete sentences. Explanatory tables and/or diagrams are acceptable, but there must always be a written discussion as well.

Submit your file to the Curator system by the posted deadline for this assignment. No late submissions will be accepted.

Lamport's Bakery Algorithm is a software-only attempt to achieve mutual exclusion and avoid deadlock; that is, the implementation does not require any system-level features like a test-and-set instruction. It is based upon a protocol used in a number of settings, including bakeries, in which every customer receives a numbered ticket upon arrival, allowing each to be served in turn. After a customer has been served, s/he discards the ticket.

```
// Globals.  n is the number of processes.  Choosing[] is initialized to
// false and Number[] is initialized to 0.
bool Choosing[n];                                                    //  1
int  Number[n];                                                      //  2

// Each process executes the following code, with a different value of
// the parameter i, in the range 0 to n-1.
P( int i ) {                                                         //  3

   while ( true ) {                                                  //  4

      Choosing[i] = true;                                            //  5
      Number[i]   = 1 + maxOf(Number[], n);                          //  6
      Choosing[i] = false;                                           //  7

      for ( int j = 0; j < n; j++) {                                 //  8
         while ( Choosing[j] );                                      //  9
         while ( Number[j] != 0 && (Number[j], j) < (Number[i], i) ); // 10
      }

      // Critical section would be here
      Number[i] = 0;                                                 // 11
      // Other code, not involving shared resources, might go here
   }
}
```

The notation `(a,b) < (c,d)` used in line 10 is defined as: `(a < c) || (a == c && b < d)`.

1. [12 points] Consider lines 5 through 7. Is the code in lines 5 and 7 necessary? If so, explain what it accomplishes.

2. [8 points] Is it possible for two different processes `P(r)` and `P(s)` where `r != s`, to have the same ticket number at the same time? In other words, is it possible for `Number[r] == Number[s]` unless both are zero? Explain.

3. [10 points] Give a brief, accurate description of the logical purpose of line 9. Your answer should be suitable to use as a comment for that statement in the algorithm.

4. [10 points] What is the largest value the loop counter `j` can reach before process `P(i)` is guaranteed that it will exit the busy-wait loop in line 9? Why?

5. [10 points] Give a brief, accurate description of the logical purpose of line 10. Your answer should be suitable to use as a comment for that statement in the algorithm.

Note: Lamport's Bakery Algorithm does, in fact, achieve mutual exclusion and avoid deadlock.

2. Consider a system that uses the Bankers' Algorithm to manage a single resource category R. The system has 150 units of R, and there are currently four processes in the system that will potentially use units of R. The current allocation state is reflected by:

Process	Current allocation	Maximum allocation
P0	40	60
P1	30	110
P2	50	130
P3	10	50

a) [15 points] According to the Bankers' Algorithm, is this a safe state or not? If yes, give a potential order of termination that can be guaranteed. If not, carefully explain why not.

b) [5 points] Are some or all of the four processes in the system described above currently in a deadlocked state? If not, is it inevitable that a deadlock will eventually occur involving some or all of these four processes?

3. [15 points] Consider the same logical scenario as in the previous question, but the current allocation state is:

Process	Current allocation	Maximum allocation
P0	50	60
P1	10	110
P2	20	130
P3	30	50

Suppose that process P2 now requests 30 units of resource R. If the request is granted, is the resulting state safe? If yes, give a potential order of termination that can be guaranteed. If not, carefully explain why not.

4. [15 points] Consider the same logical scenario as in the previous question, but the current allocation state is:

Process	Current allocation	Maximum allocation
P0	50	60
P1	10	110
P2	20	130
P3	30	50

Suppose that a fifth process is created, with a potential claim for *C* units of R and an initial need for *I* units of R. What is the maximum value of *I* for which the resulting state is safe? Explain.