1) Cache coherence protocols:
   a. Consider a system consisting of two processors, each with a separate cache. A 4-byte integer $k$ maps onto cache block $j$, the size of the cache block is 16 bytes (= four 4-byte integers). Initially, the same cache block is held in the cache of both processors. Assume now, that CPU 0 is modifying the integer $k$, while CPU 1 is trying to read element the next element of the cache block, denoted here as $k+1$, shortly after CPU 0 modified the integer $k$. Will the read operation lead to a true sharing miss in the **local cache 1** or a false sharing miss? Give a brief explanation of your answer.

   *It is a false sharing miss, since the data item that has been requested by CPU 1 has actually not been modified by CPU 0. It was just 'unlucky' to be in the same cache block as one of the data items that have been modified.*
b. Using the example snoopy protocol, we would like to track the state of block \( j \) in the local cache 0 and local cache 1 for the access pattern described above. For this, we assume that the initial state of the cache block \( j \) is SHARED in both caches. Please denote also in the last column who generated the event from the perspective of the corresponding local cache (CPU or BUS).

<table>
<thead>
<tr>
<th>Step</th>
<th>Prev. state of Cache 0 for block ( j )</th>
<th>New state of Cache 0 for block ( j )</th>
<th>Originator of event (CPU or BUS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 0 writes ( k )</td>
<td>SHARED</td>
<td><strong>EXCLUSIVE</strong></td>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td>CPU 1 reads ( k+1 )</td>
<td><strong>EXCLUSIVE</strong></td>
<td>SHARED</td>
<td><strong>BUS</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Prev. state of Cache 1 for block ( j )</th>
<th>New state of Cache 1 for block ( j )</th>
<th>Originator of event (CPU or BUS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU 0 writes ( k )</td>
<td>SHARED</td>
<td><strong>INVALID</strong></td>
<td><strong>BUS</strong></td>
</tr>
<tr>
<td>CPU 1 reads ( k+1 )</td>
<td><strong>INVALID</strong></td>
<td>SHARED</td>
<td><strong>CPU</strong></td>
</tr>
</tbody>
</table>
2.) Cache coherence protocols
   a. Given the following code sequence.

   ```c
   for ( i=0; i< N; i++) {
       a[i] = i;
   }
   ```

   Assuming that the cache is large enough to hold the entire array `a[]`, no element of `a[]` is in the cache before executing the loop, `a[]` is an array of 8 byte double precision floating point numbers, and a cache block is 64 bytes, how many cache misses will this loop create for N=1024?

   
   \[ a[0] : \text{cache miss, } a[1]-a[7] \text{ cache hit} \]
   
   \[ \text{i.e. } 1 \text{ miss for every 8 elements} \]
   
   \[ \rightarrow \text{number of cache misses } = \frac{1024}{8} = 128 \]

   b. Assuming that two threads are used to execute a code on two separate processors. Each processor has its own cache (one level) large enough to hold the entire array `a[]` with a cache block size of 64 bytes. The work distribution of threads is such that they split the loop in a round robin fashion, e.g. thread 0 on processor 0 is executing loop \( i=0, 2, 4, 6... \) and thread 1 on processor 1 executes the iteration \( i=1,3,5,7... \)

   Assuming that the caches of the two processors are using the example snoopy protocol discussed in the class, we would like to analyze an arbitrary access sequences for the variable `a[]` between the threads. Please give the cache state on each processor as well as the bus signal and the generator of the signal. Since `a[]` is not in the cache before executing the loop, you can assume that the according cache block is in the INVALID state before the loop.

<table>
<thead>
<tr>
<th>P0 cache state</th>
<th>P1 cache state</th>
<th>Bus Action</th>
<th>Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXCLUSIV</td>
<td>INVALID</td>
<td>wr miss</td>
<td>P0</td>
</tr>
<tr>
<td>INVALID</td>
<td>EXCLUSIV</td>
<td>wr back</td>
<td>P0</td>
</tr>
<tr>
<td>EXCLUSIV</td>
<td>INVALID</td>
<td>wr miss</td>
<td>P0</td>
</tr>
<tr>
<td>INVALID</td>
<td>EXCLUSIV</td>
<td>wr back</td>
<td>P1</td>
</tr>
<tr>
<td>INVALID</td>
<td>EXCLUSIV</td>
<td>wr miss</td>
<td>P1</td>
</tr>
<tr>
<td>INVALID</td>
<td>EXCLUSIV</td>
<td>wr back</td>
<td>P0</td>
</tr>
</tbody>
</table>

(2 Pts)
c. The interleaving of the two threads w.r.t. accessing the variable a[] occurs arbitrarily. The worst case scenario is given if the actual access of the variable a[] is occurring in a round robin fashion. For example, thread 0 writes a[0], thread 1 writes a[1], thread 0 writes a[2], thread 1 writes a[3] etc. How many cache misses will the code sequence of part a) generate due to the cache coherence protocol in this scenario for N=1024?

* per cache line: 1st cache miss on proc 0 is a compulsory miss
  every subsequent element in the cache line is a coherence miss
  1st cache miss on proc 1 is a compulsory miss
  every subsequent element in the cache line is a coherence miss
→ per cache line: 2 compulsory misses, 6 coherence misses
→ total number of coherence misses: 6 * 128 = 768

D. The best scenario is given if a thread can write the values of a[] into a cache block before the other thread writes its element in the same cache block. For example, thread 0 writes a[0],a[2],a[4],a[6] before thread 1 writes a[1],a[3],a[5],a[7]. How many cache misses will the code sequence above generate in this scenario for N=1024 due to the cache coherence protocol?

* per cache line: 1st cache miss on proc 0 is compulsory miss
  no subsequent cache misses for this cache line
  1st cache miss on proc 1 is a compulsory miss
  no subsequent cache misses for this cache line
→ per cache line: 2 compulsory misses but 0 coherence misses
→ total number of coherence misses: 0
2) Cache coherence protocols
   Given a NUMA system with four processors P0 - P3
   a. Explain the advantages of using a directory cache coherence protocol over a
      snoopy cache coherence protocol for NUMA architectures.
      - No bus and don’t want to broadcast:: interconnect no longer single
        arbitration point
      - Distribute directory entries with memory, each keeping track of which
        Procs have copies of their blocks
   b. Given the following configuration for the NUMA architecture above for a
      memory address x.

      | P0   | P1   | P2   | P3   |
      |------|------|------|------|
      | INVALID | EXCLUSIV | INVALID | INVALID |

      Assuming that P0 is the home node of the address x, give the list of messages
      (including sender node and receiving node) in the order that they have to
      occur for a write operation of Processor P3 to the same memory address x
      using the example directory protocol discussed in the class.

      write miss P3 -> P0
      fetch/invalidate P0 -> P1
      data write back P1 -> P0
      data value reply P0 -> P3