Midterm Solutions: CS425 FA16

1. (Solution and Grading by: Bo Teng)
   1. c
   2. c
   3. f
   4. e
   5. a
   6. c
   7. d
   8. d
   9. c
   10. c

2. (Solution and Grading by: Si Liu)
   (a)
   (b) Concurrent events with the same timestamp: S1 and S2. Concurrent events with different timestamps: S2 and the send of M6 at process P1. Other correct answers will also be accepted.

3. (Solution and Grading by: Hongwei Wang)
   (a) 89 -> 144, 144 -> 233, 233 -> 377, 377 -> 610, 610 -> 987, 987 -> 89
   (b) 987, 987, 987, 987, 987, 987, 987, 987, 987, 144
   (c) 610 -> 144 -> 233

4. (Solution and Grading by: Sili Hui)
This solution has three MapReduces chained one after another. The first calculates the count for each URL, the second extracts the top 10 URLs and corresponding IPs, and the third sorts these IPs.

Map1 (timestamp, IP, URL):
   Emit (key=URL, value=IP)

Reduce1 (key=URL, value=IP):
   Count the number of IPs for that URL, call this count_IP
   For each IP (sent with this URL)
       Emit (key=(count_IP, URL), value=IP) // Other key, value pairs may also be ok

Map2(key=(count_IP, URL), value=IP):
   Identity// Emit to a fixed key

Reduce2 // A Single reducer
   Make a pass through input list to extract (count_IP, URL) pairs
   Sort this list of (count_IP, URL) by decreasing order of count_IP, using URL (lexicographic) to break ties
       // Needed to distinguish two different URLs with the same count_IP
       // This list is small as the number of URLs is small, so sorting is efficient
   Make another pass through the sorted (count_IP, URL) list to get top 10 highest counts, while also eliminating duplicates - call this the “top 10 list”
   Make a second pass through input list to extract only those IPs that have matching (count_IP, URL) fields with top 10 list
       Emit all these IPs as (key=IP, value=_ ) // value can be anything, URL if you wish
       // At this point, it is ok for you to say sort this list but hey, we could instead use Mapreduce for that!

// Third optional stage
Mapreduce3 // sort list of IPs using Mapreduce implicit sort
   Map3(key=IP, value=_):
       Identity // Quicksort will sort the output of each map task by IP

   // Use range partitioning for shuffling by key=IP
   Reduce3(key=IP, value=_):
Eliminate duplicates and output only one (key=IP, value=(_)
                   // Reduce implicit mergesort will sort input, and therefore output
will also be sorted by key=IP

The above solution checks for duplicates (URLs with the same count), but students who ignored
this (and got the solution right otherwise) still got full points.

5. (Solution and Grading by: Shiv Verma)
(a) This is a heartbeating protocol that can provide completeness up to \( k \) failures, i.e., with
    \((k-1)\) or fewer failures, the next failure will be detected. This is because in the worst case
    the next failing \((k-\text{th failing})\) process will have at least one of its \( k \) send-to list processes
    alive to time out on its heartbeats.

(b) Since the heartbeating protocol can tolerate up to \( k \) failures, completeness (with any
    number of failures) can only be provided as long as the group size \( N \) is no more than
    \( k+1 \). In reality, to be able to select \( k \) heartbeat targets, you need at least \( k+1 \) nodes.
    However, we have given full points to all answers of \( k+1 \) or less.

(c) No heartbeating protocol is accurate in an asynchronous system, since it guarantees
    completeness (under assumptions of (a)) and we know that . Since heartbeats may be
    dropped or non-faulty processes may be late/slow in sending heartbeats, mistaken
    detections cannot be avoided.