ECE428 HW3

Q1.

a) elif message.contents == “Election”:
   # fill in the rest
      self.leader = self.pid # Assign the new leader
      for pid in self.group:
         if pid <self.pid:
        #Send Coor msg to all lower-number processes
          unicast (self.pid, “Coordinator”)

b) P4 will send election msg to all the process with second highest PID P7, and P7 sends Coordinator msg to all of the lower-numbered processes(P1-P6). Hence, totally 1+6=7 msg

c) P4 send election to P7 first within T, then P7 send to P1-P6 Coordinator msg simultaneously within T. Hence, totally T+T=2T

d) P4 will still send election msg to P7 since the failure of P7 has not been detected. Within time out P4 does not get response from P7 so it sends election msg to P6. Then P6 send Coordinator msg to P1-P5. Hence totally 2+5 = 7

e) P4 send election msg to P7 and does not get response within T+T = 2T (Time out should be 2T for a round of msg transmission). Then P4 send election msg to P6 within T. And P6 send Coordinator msg to P1-P5 simultaneously within T. Hence, totally 2T+T+T=4T

f) Best case: exactly P7 detect P8’s failure and send Coordinator msg to the lower with T.
   Worst case: exactly P1 detect P8’s failure while P7-P2 failed while P1 initiate the election. Then 2T * 6 = 12T is needed.

Q2.

a) i) No. A counterexample could be that there exist two processes carry the same interger zinput (i.e. xk = xk+1) and in this scenario it will mistakenly take xk as the max value since it think that xk is pass through the whole loop while actually it is just passed to the next process.

   ii) The judgement that xk is pass through a whole loop can not just be the value of it. Instead, there should be something record whether process pi has been checked. i.e Initialize a sent p_check as a blank set, when pi pass msg (PROPOSAL, X_K, pi), p_check.append(pi). Then change the judgement else #b==X_K into elif pi in p_check. By doing this the safety could be guaranteed.

b) PROPOSAL = 0
DECIDED =1
self_PID = # unique for each process pi (i.e PID for pi is i)
X_K = # my input
Y_K = NONE

```
def start_consensus():
   if X_K == 1:
      send(1, PROPOSAL, self_PID)
    else: # X_K = =0
      send(-1, PROPOSAL, self_PID)

def receive_message(a, b, PID):
   if Y_K is not None:
      continue # Ignore msg after decided state
   if b == DECIDED:
      if a > 0:
            Y_K =1
      elif a<0:
            Y_K = 0
      else:
            Y_K = 2
      send(a, b, PID)
    elif b ==PROPOSAL:
      if X_K = 1:
            send(a+1, PROPOSAL, PID)
      else:
            send(a-1, PROPOSAL, PID)
```

Q3.

a) B-multicast. We have assumed that unicast channels are reliable and considered Time out. We do not care on the failure in the multicast of Query msg, so pair-to-pair B-multicast should be used.

b) 2T

c) R-multicast. In the multicast of Decision msg, the phenomenon that the send may fail after it send the Decision msg could exist, which ask the reliability of multicasting Decision msg. And we do not care on the future input so pair-to-pair B-multicast is not needed.

d) Query msg from Pi to Pj within T

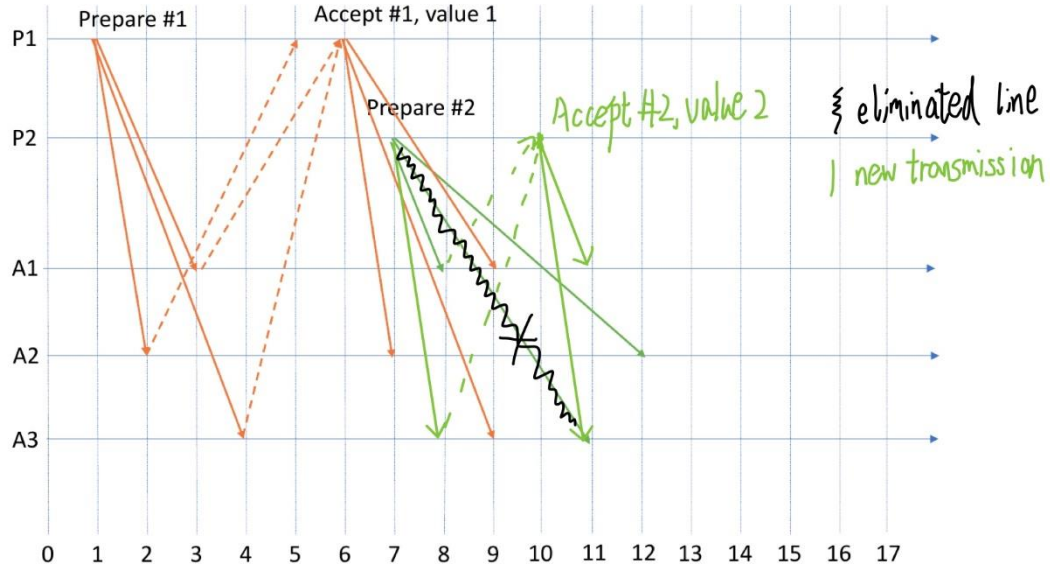
Reply msg with Xi from Each Pj to Pi within T
Decision yi from pi to each pj within T
Totally, 3T.

e) The process Pi may fail after multicasting msg to the group. And no failure detection is used to check the failure of Pi. Hence the future part of the algorithm could be meaningless and the safety could not be guaranteed for not reaching a consensus in the group.

Q4.

a) i. A1, A2, A3
ii. A2, A3
iii. P1 will send Accept msg after the Promise msg is taken after getting voted by the majority of acceptors (after receiving promise msg from Acceptor A1). Hence 3T+2T = 5T. P2 will send Accept msg after the Promise msg is taken by Acceptor A3. Hence 7T+2T = 9T.
iv. Yes, by iii we know P1 could send Accept msg to A1 and A2 at Time 5 and it take one time unit. So at Time 6 the P1 could be accepted by A1 and A2 while at Time 7 the Prepare msg just arrive at A3 (6<7). Hence, P1 could be accepted by a majority of acceptors. (2/3>1/2)

b) i. A1, A2
ii. A1, A2, A3
iii. P2 will send Accept msg after getting the Promise msg by the majority of acceptors (after A3). Hence 11T+2T =13T
iv. All of the acceptors: A1, A2, A3.
v. value 1. Since P2 has not start sending Accept msg.
vi. Change the arrived Time of Prepare msg to A3 from 11T to 8T hence P1 has not been accepted by all of the acceptors (at 9T, 8<9).



Q5.

a) P5

b) 75+5=50+30
At time 80ms P2 and P3 votes for themselves
P3 votes for P3
P2 votes for P2
P4 votes for P3 since P3's election msg arrived faster than from P2
P5 votes for P2 since P2's election msg arrived faster than from P3

c) All votes for P5
Because at 50+15 = 65, P5 begins send election msg to all the other processes. P5 votes for P5.
P4 received P5 election at 65+15 = 80 (80<20+75) so it votes for P5
P3 received P5 election at 65+20 = 85 (80<5+100) so it votes for P5
P2 received P5 election at 65+5 = 70 (80<30+150) so it votes for P5

d) P2, P3, P5, P1, P4