Pseudocode

TicketBooker

```
1  Initialize Q1(sorted by priority then arrival time then process index), Q2(sorted by arrival time then process
   index), hasNotArrived and hasEnded queues
2  QUEUE_ONE_TIME_QUOTA = 5, QUEUE_TWO_TIME_QUOTA = 20, timer = 0
3  while (Q1 ≠ ∅ AND Q2 ≠ ∅ AND hasNotArrived ≠ ∅)
4     currentProcessIndex = 0
5     while (Q1 ≠ ∅)                                                          // Process Q1 first
6       processTime = 0
7       popHead(Q1) ⇒ P                                          // Pop head of Q1 and store it into P
8       P.sinceLastRun = timer
9       while (P.totalTickets > 0 AND processTime < QUEUE_ONE_TIME_QUOTA)
10        timer++
11        P.totalTickets−−
12        processTime++
13        getHead(hasNotArrived) ⇒ R       // Get & store next process in hasNotArrived into R
   ==================INSERT NEW ARRIVAL PROCESSES FIRST==================
14        while (R.arrivalTime == timer)      // Check if there are processes that has arrived
15          if (R.priority > THRESHOLD) *****REVISE THE IF BLOCK*****    // Insert into Q1
16            processInsertIndex = 0
17            while (Q1[processInsertIndex].priority >= R.priority)          // Find correct insert
   position
18              insertIndex++
19            if (currentProcessIndex == processInsertIndex − 1) // New arrival process first
   before pre-empted process
20              insert R into Q1 before the (processInsertIndex − 1)th element of Q1
21            else
22              insert R into Q1 before the (processInsertIndex)th element of Q1
23            if (processInsertIndex >= currentProcessIndex)
24              currentProcessIndex++
25          else                                                              // Insert into Q2
26            tailInsert(R, Q2)                                    // Insert at the end of Q2
27          popHead(hasNotArrived)                                // Remove head of hasNotArrived
28          getHead(hasNotArrived) ⇒ R
   ========================================================================
29      P.running += processTime
30      P.runningInQueueOne += QUEUE_ONE_TIME_QUOTA
   ====================INSERT PRE-EMPTED PROCESS SECOND====================
31      if (P.totalTickets ≠ 0)
32        if (P.runningInQueueOne == 25)     // Check if the pre-empted process has ran for 25
   time units in Q1
33          P.priority−−
34          if (P.priority ≤ THRESHOLD)                                     // Demote process
35            P.runningInQueueOne = 0          // Reset the counter for total time ran in Q1
36            tailInsert(P, Q2)
37          else
38            tailInsert(P, Q1) *****REVISE THIS*****
```

```
39          else
40            tailInsert(P, Q1) *****REVISE THIS*****          // Insert at end of Q1 due to RR
41        else
42          tailInsert(P, hasEnded)                    // Pre-empted process has finished its job
    ============================================================
43        for every S in Q1        // Increment waiting time of a process if it has ran except the
    current running process
44          if (S.sinceLastRun ≠ -1)
45            S.waiting += processTime
    ====================INSERT PROMOTED PROCESSES THIRD====================
46        for every T in Q2                                  // Do the same in Q2 for demoted process
47          if (T.sinceLastRun ≠ -1)
48            T.waiting += processTime
49          T.age++                          // Also Increment the age of the processes in Q2 by 1
50          if (T.age == 8)
51            T.priority++
52            T.age = 0
53          if (T.priority > THRESHOLD)                    // Move the promoted process into Q1
54            tailInsert(T, Q1)
    ============================================================
55      while (Q2 ≠ ∅ AND Q1 == ∅)                                                  // Process Q2
56        popHead(Q2) ⇒ P                                 // Pop head of Q2 and store it into P
57        while (P.totalTickets > 0 AND P.remainingProcessTime > 0)
58          timer++
59          P.totalTickets−−
50          P.remainingProcessTime−−
61          getHead(hasNotArrived) ⇒ R       // Get & store next process in hasNotArrived into R
62          while (R.arrivalTime == timer)     // Check if there are processes that has arrived
63            if (R.priority > THRESHOLD)                                    // Insert into Q1
64              headInsert(R, Q1) *****REVISE THIS FOR MULTIPLE ARRIVAL*****
65            else
66              tailInsert(R, Q2)
67            popHead(hasNotArrived)                    // Remove head of hasNotArrived
68            getHead(hasNotArrived) ⇒ R
69          if (Q1 ≠ ∅)
70            headInsert(P, Q2)                         // Pre-empt the running process
71            exit while loop
```