1. Pseudocode

```
MAX_CUSTOMERS = 100
tellerCount = 0
customerCount = 0
servedCustomers = 0
maxQueueLength = 0
currentQueueLength = 0
simulationTime = 0.0000
totalServingTime = 0.0000
totalWaitingTime = 0.0000
highPriority[MAX_CUSTOMERS][3]
medPriority[MAX_CUSTOMERS][3]
lowPriority[MAX_CUSTOMERS][3]
lowPointer = 0
medPointer = 0
highPointer = 0
tellers[tellerCount][4] = 0,0,0,0

Procedure main()
     tellerCount = input number of tellers

     call Procedure loadCustomerData
     call Procedure runSimulation
end

Procedure loadCustomerData
     filename[20]

     filename = input file name
     open file
     if file not found then
          print error
     endif

     lowCounter = 0
     medCounter = 0
     highCounter = 0
     while reading lines in file get arrivalTime, serviceTime,
priority
          switch for priority
               case 1
                    insert arrivalTime, serviceTime, priority into
lowPriority[lowCounter]
                    lowCounter++
```

```
                case 2
                     insert arrivalTime, serviceTime, priority into
medPriority[medCounter]
                     medCounter++
                case 3
                     insert arrivalTime, serviceTime, priority into
highPriority[highCounter]
                     highCounter++
           end switch

           customerCount++

           if arrivalTime = 0 and serviceTime = 0 then
                break
           endif

     close file
end

Procedure runSimulation

     while servedCustomers < customerCount
           nextTeller = call Procedure getNextTeller

           nextCustomer = call Procedure getNextCustomer(nextTeller)
           if nextCustomer arrivalTime < nextTeller finishTime then
                totalWaitingTime += nextTeller finishTime -
nextCustomer arrivalTime
                maxQueueLength = call Procedure
getQueueLength(nextTeller)
           endif

           if nextTeller finishTime < nextCustomer arrivalTime then
                nextTeller idleTime += nextCustomer arrivalTime -
nextTeller finishTime
                nextTeller finishTime = nextCustomer arrivalTime +
nextCustomer serviceTime

                if currentQueueLength > maxQueueLength then
                     maxQueueLength = currentQueueLength
                endif
           else
                nextTeller finishTime += nextCustomer serviceTime
           endif
```

```
            currentQueueLength--
            nextTeller servingTime += nextCustomer serviceTime
            nextTeller customersServed++
            simulationTime = nextTeller finishTime
            totalServingTime += nextTeller servingTime
            servedCustomers++
      endwhile

      call Procedure printStatistics
end

Procedure getNextTeller
      nextTeller = 0
      for each teller
            compare other tellers
                  nextTeller = teller with earliest finish time
      endfor
      return nextTeller
end

Procedure getQueueLength(nextTeller)
      currentQueueLength++

      for each customer starting at index priority pointer in each
priority queue
            if simulationTime <= customer arrivalTime < nextTeller
finishTime then
                  currentQueueLength++
            endif
      endfor

      if currentQueueLength > maxQueueLength then
            return currentQueueLength
      else
            return maxQueueLength
      endif
end

// Identify next customer by checking the customer arrival times in
each priority queue against the next teller's finish time
Procedure getNextCustomer(nextTeller)
      customerDetails[2]

      // Checking if a customer is queued
```

```
     if highPriority[highPointer] arrivalTime < nextTeller
finishTime then
          customerDetails = highPriority[highPointer] arrivalTime
and serviceTime
          highPointer++
          return customerDetails

     elseif medPriority[medPointer] arrivalTime < nextTeller
finishTime then
          customerDetails = medPriority[medPointer] arrivalTime and
serviceTime
          medPointer++
          return customerDetails

     elseif lowPriority[lowPointer] arrivalTime < nextTeller
finishTime then
          customerDetails = lowPriority[lowPointer] arrivalTime and
serviceTime
          lowPointer++
          return customerDetails
     endif

     // If no customer in the queue, identify customer with the
earliest arrival time
     if (highPriority[highPointer] arrivalTime <
medPriority[medPointer] arrivalTime) AND (highPriority[highPointer]
arrivalTime < lowPriority[lowPointer] arrivalTime) then
          customerDetails = highPriority[highPointer] arrivalTime
and serviceTime
          highPointer++
          return customerDetails

     else if (medPriority[medPointer] arrivalTime <
highPriority[highPointer] arrivalTime) AND (medPriority[medPointer]
arrivalTime < lowPriority[lowPointer] arrivalTime) then
          customerDetails = medPriority[medPointer] arrivalTime and
serviceTime
          medPointer++
          return customerDetails

     else if (lowPriority[lowPointer] arrivalTime <
medPriority[medPointer] arrivalTime) AND (lowPriority[lowPointer]
arrivalTime < highPriority[highPointer] arrivalTime) then
          customerDetails = lowPriority[lowPointer] arrivalTime and
serviceTime
```

```
            lowPointer++
            return customerDetails
      endif
end

Procedure printStatistics
      print simulationTime
      print average service time
      print average waiting time
      print maxQueueLength
      print average queue length

      for each teller
            print customers served
            print idle rate
      endfor
end
```

2. big-O analysis

- Function getQueueLength()
  - for (int i = x; i < customerCount; i++)
  - for (int i = x; i < customerCount; i++)
  - for (int i = x; i < customerCount; i++)
  - O(1) as customerCount is an unchanged value

- Function getNextTeller()
  - for (int i = 0; i < x; i++){
    for (int k = 0; k < x; k++)
    }
  - $O(n^2)$ as there's a nested for loop

3. Data Structures

- Array
- 2D Array
  - Used to store tellers
  - Used to store each customer of different priorities

4. Compilation and Execution

```
Enter amount of tellers to run the simulation with: 1
Enter the customer file name: a2-sample.txt
Simulation Time: 1279.92
Average Service Time: 636.095
Average Waiting Time: 372.597
Max Queue Length: 56
Average Queue Length: 0.585756

----------
Teller 1
Customers Served: 100
Idle Time: 2.7388
Idle Rate: 0.00213983
----------
```

5. Outputs

```
Enter amount of tellers to run the simulation with: 1
Enter the customer file name: a2-sample.txt
Simulation Time: 1279.92
Average Service Time: 636.095
Average Waiting Time: 372.597
Max Queue Length: 56
Average Queue Length: 0.585756

----------
Teller 1
Customers Served: 100
Idle Time: 2.7388
Idle Rate: 0.00213983
----------
```

```
Enter amount of tellers to run the simulation with: 2
Enter the customer file name: a2-sample.txt
Simulation Time: 664.125
Average Service Time: 322.023
Average Waiting Time: 66.7559
Max Queue Length: 22
Average Queue Length: 0.224672

----------
Teller 1
Customers Served: 50
Idle Time: 13.2869
Idle Rate: 0.0200066
----------
Teller 2
Customers Served: 50
Idle Time: 24.301
Idle Rate: 0.036591
----------
```

```
Enter amount of tellers to run the simulation with: 4
Enter the customer file name: a2-sample.txt
Simulation Time: 0
Average Service Time: 0
Average Waiting Time: 0
Max Queue Length: 0
Average Queue Length: nan

----------
Teller 1
Customers Served: 0
Idle Time: 0
Idle Rate: nan
----------
Teller 2
Customers Served: 0
Idle Time: 0
Idle Rate: nan
----------
Teller 3
Customers Served: 0
Idle Time: 0
Idle Rate: nan
----------
Teller 4
Customers Served: 0
Idle Time: 0
Idle Rate: nan
----------
```

6. Statistic Discussion

The more tellers that are available will result in a smaller average waiting time per customer, and also a shorter average servicing time as multiple customers can be served at one time. The simulation time will also be reduced as the customers can be served more efficiently. The max queue length will also be reduced, along with the average queue length.

The teller's idle time is increased as a result of fewer customers being in the queue.