Variables

```
BLANK = "  "
MAX_Q_SIZE = 30
MAX_TILLS = 5
MAX_TIME = 50
TILL_SPEED = 3

TIME_IDLE = 0
TIME_BUSY = 1
TIME_SERVING = 2

ARRIVAL_TIME = 0
ITEMS = 1

# indices for Stats data structure
MAX_Q_LENGTH = 0
MAX_WAIT = 1
TOTAL_WAIT = 2
TOTAL_Q = 3
TOTAL_Q_OCCURRENCE = 4
TOTAL_NO_WAIT = 5
```

```
class Q_Node:
  def __init__(self):
    self.BuyerID = BLANK
    self.WaitingTime = 0
    self.ItemsInBasket = 0
```

```
def ResetDataStructures():
  Stats = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
  Tills = [[0, 0, 0] for i in range(MAX_TILLS + 1)]
  BuyerQ = [Q_Node() for i in range(MAX_Q_SIZE)]
  return Stats, Tills, BuyerQ
```

```
def ServeBuyer(BuyerQ, QLength):
  ThisBuyerID = BuyerQ[0].BuyerID
  ThisBuyerWaitingTime = BuyerQ[0].WaitingTime
  ThisBuyerItems = BuyerQ[0].ItemsInBasket
  for Count in range(QLength):
    BuyerQ[Count].BuyerID = BuyerQ[Count + 1].BuyerID
    BuyerQ[Count].WaitingTime = BuyerQ[Count + 1].WaitingTime
    BuyerQ[Count].ItemsInBasket = BuyerQ[Count + 1].ItemsInBasket
  BuyerQ[QLength].BuyerID = BLANK
  BuyerQ[QLength].WaitingTime = 0
  BuyerQ[QLength].ItemsInBasket = 0
  QLength -= 1
  print(f"{ThisBuyerID:>17s}", end='')
  return BuyerQ, QLength, ThisBuyerID, ThisBuyerWaitingTime, ThisBuyerItems
```

```
def ReadInSimulationData():
  Data = [[0, 0] for i in range(MAX_TIME + 1)]
  FileIn = open("SimulationData.txt", 'r')
  DataString = FileIn.readline()
  Count = 0
  while DataString != "" and Count < MAX_TIME:
    Count += 1
    Data[Count][ARRIVAL_TIME] = int(DataString[0])
    Data[Count][ITEMS] = int(DataString[2:])
    DataString = FileIn.readline()
  FileIn.close()
  return Data
```

Main Function

```
def QueueSimulator():
  BuyerNumber = 0
  QLength = 0
  Stats, Tills, BuyerQ = ResetDataStructures()
  SimulationTime, NoOfTills = ChangeSettings()
  Data = ReadInSimulationData()
  OutputHeading()
  TimeOfNextArrival = Data[BuyerNumber + 1][ARRIVAL_TIME]
  for TimeUnit in range(SimulationTime):
    TimeToNextArrival -= 1
    print(f"{TimeUnit:>3d}", end='')
    if TimeToNextArrival == 0:
      BuyerNumber += 1
      BuyerQ, QLength, NoOfTills, Stats = BuyerArrives(Data, BuyerQ, QLength, BuyerNumber, NoOfTills, Stats)
      TimeToNextArrival = Data[BuyerNumber + 1][ARRIVAL_TIME]
    else:
      print()
    Tills, NoOfTills, BuyerQ, QLength, Stats = Serving(Tills, NoOfTills, BuyerQ, QLength, Stats)
  ExtraTime = 0
  while QLength > 0:
    TimeUnit = SimulationTime + ExtraTime
    print(f"{TimeUnit:>3d}")
    Tills, NoOfTills, BuyerQ, QLength, Stats = Serving(Tills, NoOfTills, BuyerQ, QLength, Stats)
    ExtraTime += 1
  while TillsBusy(Tills, NoOfTills):
    TimeUnit = SimulationTime + ExtraTime
    print(f"{TimeUnit:>3d}")
    Tills = UpdateTills(Tills, NoOfTills)
    OutputTillAndQueueStates(Tills, NoOfTills, BuyerQ, QLength)
    ExtraTime += 1
  OutputStats(Stats, BuyerNumber, SimulationTime)
```

```
def ChangeSettings():
  SimulationTime = 10
  NoOfTills = 2
  print("Settings set for this simulation:")
  print("================================")
  print(f"Simulation time: {SimulationTime}")
  print(f"Tills operating: {NoOfTills}")
  print("================================")
  print()
  Answer = input("Do you wish to change the settings?  Y/N: ")
  if Answer == 'Y':
    print(f"Maximum simulation time is {MAX_TIME} time units")
    SimulationTime = int(input("Simulation run time: "))
    while SimulationTime > MAX_TIME or SimulationTime < 1:
      print(f"Maximum simulation time is {MAX_TIME} time units")
      SimulationTime = int(input("Simulation run time: "))
    print(f"Maximum number of tills is {MAX_TILLS}")
    NoOfTills = int(input("Number of tills in use: "))
    while NoOfTills > MAX_TILLS or NoOfTills < 1:
      print(f"Maximum number of tills is {MAX_TILLS}")
      NoOfTills = int(input("Number of tills in use: "))
  return SimulationTime, NoOfTills
```

```
def UpdateStats(Stats, WaitingTime):
  Stats[TOTAL_WAIT] += WaitingTime
  if WaitingTime > Stats[MAX_WAIT]:
    Stats[MAX_WAIT] = WaitingTime
  if WaitingTime == 0:
    Stats[TOTAL_NO_WAIT] += 1
  return Stats
```

```
def OutputHeading():
  print()
  print("Time Buyer  | Start Till Serve | Till Time Time Time |      Queue")
  print("    enters | serve      time  | num- idle busy ser- | Buyer Wait Items")
  print("    (items)| buyer            | ber            ving | ID    time in")
  print("           |                  |                     |             basket")
```

```
def CalculateServingTime(Tills, ThisTill, NoOfItems):
  ServingTime = (NoOfItems // TILL_SPEED) + 1
  Tills[ThisTill][TIME_SERVING] = ServingTime
  print(f"{ThisTill:>6d}{ServingTime:>6d}")
  return Tills
```

```
def FindFreeTill(Tills, NoOfTills):
  FoundFreeTill = False
  TillNumber = 0
  while not FoundFreeTill and TillNumber < NoOfTills:
    TillNumber += 1
    if Tills[TillNumber][TIME_SERVING] == 0:
      FoundFreeTill = True
  if FoundFreeTill:
    return TillNumber
  else:
    return -1
```

```
def Serving(Tills, NoOfTills, BuyerQ, QLength, Stats):
  TillFree = FindFreeTill(Tills, NoOfTills)
  while TillFree != -1 and QLength > 0:
    BuyerQ, QLength, BuyerID, WaitingTime, ItemsInBasket = ServeBuyer(BuyerQ, QLength)
    Stats = UpdateStats(Stats, WaitingTime)
    Tills = CalculateServingTime(Tills, TillFree, ItemsInBasket)
    TillFree = FindFreeTill(Tills, NoOfTills)
  BuyerQ = IncrementTimeWaiting(BuyerQ, QLength)
  Tills = UpdateTills(Tills, NoOfTills)
  if QLength > 0:
    Stats[TOTAL_Q_OCCURRENCE] += 1
    Stats[TOTAL_Q] += QLength
  if QLength > Stats[MAX_Q_LENGTH]:
    Stats[MAX_Q_LENGTH] = QLength
  OutputTillAndQueueStates(Tills, NoOfTills, BuyerQ, QLength)
  return Tills, NoOfTills, BuyerQ, QLength, Stats
```

```
def BuyerArrives(Data, BuyerQ, QLength, BuyerNumber, NoOfTills, Stats):
  print(f"  B{BuyerNumber}{Data[BuyerNumber][ITEMS]:>7d}")
  BuyerQ, QLength = BuyerJoinsQ(Data, BuyerQ, QLength, BuyerNumber)
  return BuyerQ, QLength, NoOfTills, Stats
```

```
def BuyerJoinsQ(Data, BuyerQ, QLength, BuyerNumber):
  ItemsInBasket = Data[BuyerNumber][ITEMS]
  BuyerQ[QLength].BuyerID = f"B{BuyerNumber}"
  BuyerQ[QLength].ItemsInBasket  = ItemsInBasket
  QLength += 1
  return BuyerQ, QLength
```

```
def OutputStats(Stats, BuyerNumber, SimulationTime):
  print("The simulation statistics are:")
  print("==============================")
  print(f"The maximum queue length was: {Stats[MAX_Q_LENGTH]} buyers")
  print(f"The maximum waiting time was: {Stats[MAX_WAIT]} time units")
  print(f"{BuyerNumber} buyers arrived during {SimulationTime} time units")
  AverageWaitingTime = round(Stats[TOTAL_WAIT] / BuyerNumber, 1)
  print(f"The average waiting time was: {AverageWaitingTime} time units")
  if Stats[TOTAL_Q_OCCURRENCE] > 0:
    AverageQLength = round(Stats[TOTAL_Q] / Stats[TOTAL_Q_OCCURRENCE], 2)
    print(f"The average queue length was: {AverageQLength} buyers")
  print(f"{Stats[TOTAL_NO_WAIT]} buyers did not need to queue")
```

```
def OutputTillAndQueueStates(Tills, NoOfTills, BuyerQ, QLength):
  for i in range(1, NoOfTills + 1):
    print(f"{i:>36d}{Tills[i][TIME_IDLE]:>5d}{Tills[i][TIME_BUSY]:>5d}{Tills[i][TIME_SERVING]:>6d}")
  print("                                        ** Start of queue **")
  for i in range(QLength):
    print(f"{BuyerQ[i].BuyerID:>57s}{BuyerQ[i].WaitingTime:>7d}{BuyerQ[i].ItemsInBasket:>6d}")
  print("                                        *** End of queue ***")
  print("-------------------------------------------------------------")
```

```
def TillsBusy(Tills, NoOfTills):
  IsBusy = False
  TillNumber = 0
  while not IsBusy and TillNumber <= NoOfTills:
    if Tills[TillNumber][TIME_SERVING] > 0:
      IsBusy = True
    TillNumber += 1
  return IsBusy
```

```
def UpdateTills(Tills, NoOfTills):
  for TillNumber in range(NoOfTills + 1):
    if Tills[TillNumber][TIME_SERVING] == 0:
      Tills[TillNumber][TIME_IDLE] += 1
    else:
      Tills[TillNumber][TIME_BUSY] += 1
      Tills[TillNumber][TIME_SERVING] -= 1
  return Tills
```

```
def IncrementTimeWaiting(BuyerQ, QLength):
  for Count in range(QLength):
    BuyerQ[Count].WaitingTime += 1
  return BuyerQ
```

Type to enter text