Computing and Data Science

Simulations

Assignment no. 1 code

3rd Year

ID: 20221449583

Name: Ali Mohamed Sayed Ahmed

Eng.Mohamed Hatem

Dr. Emad Rauf

Screen of code:

```
[2]  import numpy as np
     import pandas as pd
     import os
     import queue
     import random
     import csv
```

```
class task:
    totalWaitingTime, numOftasks = 0, 0  # Class variables for total waiting time and number of tasks
    def csv(self):  # Method to convert task data to CSV format
        data = [
            str(self.id), str(self.IAT), str(self.AT), str(self.ASTBegin), str(self.AST),
            str(self.ASTEnds), str(self.BSTBegin), str(self.BST), str(self.BSTEnds),
            str(self.WT), str(self.timespend), str(self.idle)
        ]
        return data
    def __init__(self):  # Constructor method to initialize task attributes
        id, IAT, AST, AT, WT, timespend, ASTBegin, ASTEnds, idle = 0, 0, 0, 0, 0, 0, 0, 0, 0
        BST, BSTBegin, BSTEnds = 0, 0, 0  # Initialize variables to 0
        self.numOftasks += 1  # Increment the number of tasks when a new task is created
    def IATgenerator(self, RN, m):  # Method to generate Inter-Arrival Time (IAT)
        r = RN[m]  # Get random number from RN list
        IAT = 0  # Initialize IAT

        # Determine IAT based on ranges of random number
        if r >= 0 and r <= 24:
            IAT = 1
        elif r >= 25 and r <= 64:
            IAT = 2
        elif r >= 65 and r <= 84:
            IAT = 3
        elif r >= 85 and r <= 99:
            IAT = 4
        return IAT
```

Connected to Python 3 Google Compu

```
[5]  class able:
         state = 0  # State of the able
         AbleST = 0  # Able's service time
         currentST = 0  # Current service time
         totalServicesTime = 0  # Total service time

         def serviceTimeGeneretorAble(self, RM, m):  # Method to generate service time for Able
             r = RM[m]  # Get random number from RM list
             serviceTime = 0  # Initialize service time

             # Determine service time based on ranges of random number
             if r >= 0 and r <= 29:
                 serviceTime = 2
             elif r >= 30 and r <= 57:
                 serviceTime = 3
             elif r >= 58 and r <= 82:
                 serviceTime = 4
             elif r >= 83 and r <= 99:
                 serviceTime = 5

             return serviceTime
```

```
[24] class baker:
         state = 0  # State of the baker
         BakerST = 0  # Baker's service time
         currentST = 0  # Current service time
         totalServicesTime = 0  # Total service time

         def serviceTimeGeneretorBaker(self, RN, m):  # Method to generate service time for Baker
             r = RN[m]  # Get random number from RN list
             serviceTime = 0  # Initialize service time

             # Determine service time based on ranges of random number
             if r >= 0 and r <= 34:
                 serviceTime = 2
             elif r >= 35 and r <= 59:
                 serviceTime = 3
             elif r >= 60 and r <= 79:
                 serviceTime = 4
             elif r >= 80 and r <= 99:
                 serviceTime = 5

             return serviceTime
```

```
✓ 4s [28] Able = able()  # Creating an instance of the 'able' class
         Baker = baker()  # Creating an instance of the 'baker' class
         taskQueue = queue.Queue()  # Creating a queue to hold tasks
         taskInfo = []  # List to hold task information

         # Initializing variables
         nextAt, nextIAT, nextWT, totalWT, numOftasksWhoWaited, tempforidleBaker, tempforidleAble = 0, 0, 0, 0, 0, 0, 0

         # User input for simulation time and choice of random numbers
         simulationTime = int(input("Enter simulation time: "))
         choice = int(input("Enter 1 to use the same assignment's numbers or Enter 2 to use random numbers from the table: "))

         # Assigning random numbers based on user's choice
         if choice == 1:
             RN = [26, 95, 98, 21, 90, 51, 26, 92, 42, 89, 74, 38, 80, 13, 68, 61,
                   22, 50, 48, 49, 34, 39, 45, 53, 24, 88, 34, 1, 63, 81, 38, 53, 80, 81, 42,
                   64, 56, 1, 89, 67, 18, 1, 51, 47, 71, 75, 16, 57, 92, 87, 47, 16, 98, 34, 21, 96, 90]
             m = 0
         elif choice == 2:
             RN = [94, 73, 70, 82, 25, 35, 61, 42, 48, 26, 88, 31, 90, 55, 95, 58, 70,
                   15, 73, 65, 74, 75, 98, 72, 59, 85, 98, 21, 32, 96, 89,
                   32, 67, 48, 63, 99, 98, 66, 85, 58, 6, 39, 15, 2, 48, 63,
                   85, 61, 40, 16, 18, 52, 71, 16, 34, 96, 90, 85, 81, 46, 53, 49, 65, 17, 30, 3, 50]
             m = random.randint(0, 10)

         id = 1  # Initializing task ID

         # Simulation loop
         for clock in range(0, simulationTime) or queue.size > 0:
             # Creating new tasks if it's their arrival time
             if clock == nextAt:
                 newtask = task()
                 newtask.id = id
                 newtask.AT = nextAt
                 newtask.IAT = nextIAT
                 taskQueue.put(newtask)
```

✓ 4s completed at 10:50 PM

```
[28]
            nextIAT = newtask.IATgenerator(RN, m)
            m += 1
            if nextAt + nextIAT < simulationTime:
                nextAt = nextIAT + nextAt
            id += 1

        # Processing tasks by Baker
        if Baker.state == 0:
            if not taskQueue.empty():
                Baker.state = 1
                removedOfQueue = taskQueue.get()
                removedOfQueue.BST = Baker.serviceTimeGeneretorBaker(RN, m)
                m += 1
                removedOfQueue.WT = clock - removedOfQueue.AT
                removedOfQueue.timespend = removedOfQueue.BST + removedOfQueue.WT
                removedOfQueue.BSTBegin = clock
                removedOfQueue.idle = removedOfQueue.BSTBegin - tempforidleBaker
                removedOfQueue.BSTEnds = clock + removedOfQueue.BST
                removedOfQueue.AST, removedOfQueue.ASTBegin, removedOfQueue.ASTEnds = " ", " ", " "
                tempforidleBaker = removedOfQueue.BSTEnds
                Baker.currentST = removedOfQueue.BST
                Baker.totalServicesTime += removedOfQueue.BST
                taskInfo.append(removedOfQueue)
        else:
            Baker.currentST -= 1
            if Baker.currentST == 1:
                Baker.state = 0

        # Processing tasks by Able
        if Able.state == 0:
            if not taskQueue.empty():
                Able.state = 1
                removedOfQueue = taskQueue.get()
                removedOfQueue.AST = Able.serviceTimeGeneretorAble(RN, m)
                m += 1
                removedOfQueue.WT = clock - removedOfQueue.AT
```

```
[28]
                removedOfQueue.timespend = removedOfQueue.AST + removedOfQueue.WT
                removedOfQueue.ASTBegin = clock
                removedOfQueue.idle = removedOfQueue.ASTBegin - tempforidleAble
                removedOfQueue.ASTEnds = clock + removedOfQueue.AST
                removedOfQueue.BST, removedOfQueue.BSTBegin, removedOfQueue.BSTEnds = " ", " ", " "
                tempforidleAble = removedOfQueue.ASTEnds
                Able.currentST = removedOfQueue.AST
                Able.totalServicesTime += removedOfQueue.AST
                taskInfo.append(removedOfQueue)
        else:
            Able.currentST -= 1
            if Able.currentST == 1:
                Able.state = 0

Enter simulation time: 60
Enter 1 to use the same assignment's numbers or Enter 2 to use random numbers from the table: 1
```

```python
csv_file = []  # List to hold CSV data
path = 'simulation.csv'  # File path for CSV file

# Check if the file already exists, if so, remove it
if os.path.exists(path) and os.path.isfile(path):
    os.remove(path)

# Open the file in write mode
file = open(path, 'w', newline='')
writer = csv.writer(file)

# Write header row for the CSV file
writer.writerow(['id', 'IAT', 'AT', 'A-ST begin', 'A-ST', 'A-ST ends', 'B-ST begin', 'B-ST', 'B-ST ends', 'WT', 'Time spend', 'idle time'])

# Write task information to the CSV file
for i in range(0, len(taskInfo)):
    writer.writerow(taskInfo[i].csv())  # Writing CSV data for each task
    if taskInfo[i].WT > 0:
        numOftasksWhoWaited += 1  # Increment count of tasks that waited
        totalWT = totalWT + taskInfo[i].WT  # Accumulate total waiting time

file.close()  # Close the CSV file

# Calculate and print statistics
print("Able busy time percentage to total time", (Able.totalServicesTime / nextAt) * 100)
print("Baker busy time percentage to total time", (Baker.totalServicesTime / nextAt) * 100)
print("Average waiting Time =", numOftasksWhoWaited / totalWT)
```

```
Able busy time percentage to total time 66.10169491525424
Baker busy time percentage to total time 86.4406779661017
Average waiting Time = 1.0
```