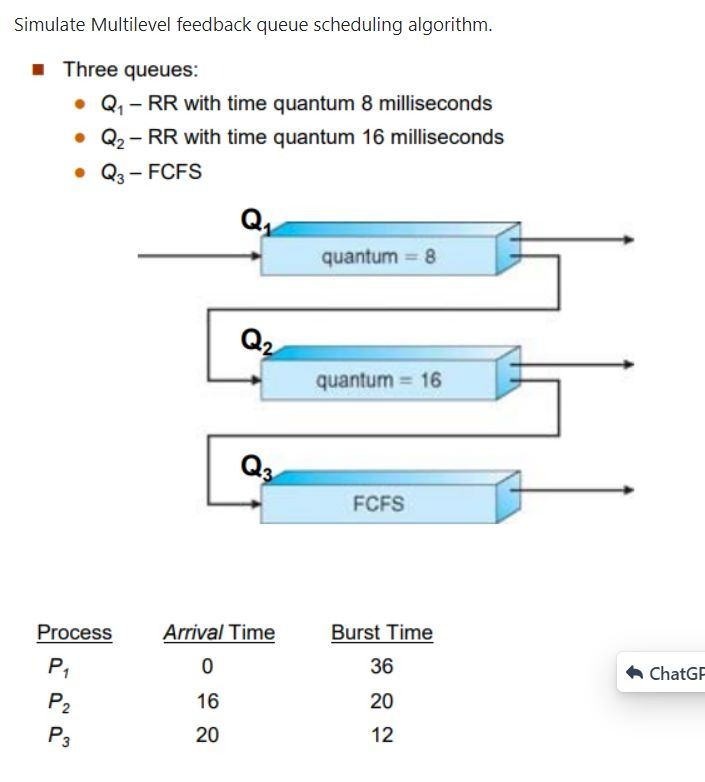
Name: Mudit Mohit Assignment 7

Enrolment Number: BT21GCS057

Section: B2

Q.) 

Ans.)

Code:

class Process:

def \_\_init\_\_(self, pid, arrival\_time, burst\_time):

self.pid = pid

self.arrival\_time = arrival\_time

self.burst\_time = burst\_time

self.remaining\_time = burst\_time

self.start\_time = -1

self.completion\_time = -1

def execute(self, quantum):

if self.remaining\_time <= quantum:

time = self.remaining\_time

self.remaining\_time = 0

else:

time = quantum

self.remaining\_time -= quantum

return time

def is\_completed(self):

return self.remaining\_time == 0

def turnaround\_time(self):

return self.completion\_time - self.arrival\_time

def waiting\_time(self):

return self.turnaround\_time() - self.burst\_time

def response\_time(self):

return self.start\_time - self.arrival\_time

def relative\_delay(self):

return self.waiting\_time() / self.turnaround\_time()

def multilevel\_feedback\_queue\_scheduling(processes, quantum1=8, quantum2=16):

q1 = []

q2 = []

q3 = []

for process in processes:

q1.append(process)

current\_time = 0

while q1 or q2 or q3:

if q1:

process = q1.pop(0)

process.start\_time = current\_time

time = process.execute(quantum1)

current\_time += time

if process.is\_completed():

process.completion\_time = current\_time

print(f"{process.pid}\t{process.turnaround\_time()}\t{process.waiting\_time()}\t{process.response\_time()}\t{process.relative\_delay()}")

else:

q2.append(process)

elif q2:

process = q2.pop(0)

process.start\_time = current\_time

time = process.execute(quantum2)

current\_time += time

if process.is\_completed():

process.completion\_time = current\_time

print(f"{process.pid}\t{process.turnaround\_time()}\t{process.waiting\_time()}\t{process.response\_time()}\t{process.relative\_delay()}")

else:

q3.append(process)

elif q3:

process = q3.pop(0)

process.start\_time = current\_time

time = process.burst\_time

current\_time += time

process.completion\_time = current\_time

print(f"{process.pid}\t{process.turnaround\_time()}\t{process.waiting\_time()}\t{process.response\_time()}\t{process.relative\_delay()}")

# Input

processes = [

Process("P1", 0, 36),

Process("P2", 16, 20),

Process("P3", 20, 12),

]

# Output: TAT, WT, RT, RD.

print("Process\tTAT\tWT\tRT\tRD")

multilevel\_feedback\_queue\_scheduling(processes)

Output:

