CERTIFICATE

Name of the Lab : OPERATING SYSTEMS

Name of the Student : JASWANTH YENDURI

Student Regd. No. : 18BQ1A05M8

CLASS : III B.TECH. I SEM CSE – A

GIT HUB LINK: https://github.com/Jaswanth-yenduri/OS-Lab.git

INDEX

|  |  |  |
| --- | --- | --- |
| S. No. | Name of the Experiment | Page No. |
| 1 (a) | Simulate FCFS CPU scheduling algorithm | 3 |
| 1 (b) | Simulate Non Preemptive SJF CPU scheduling algorithm | 8 |
| 1 (c) | Simulate Preemptive SJF CPU scheduling algorithm | 14 |
| 1 (d) | Simulate Non Preemptive Priority CPU scheduling algorithm | 21 |
| 1 (e) | Simulate Preemptive Priority CPU scheduling algorithm | 29 |
| 1 (f) | Simulate RoundRobin CPU scheduling algorithm | 37 |

Z

**EXPERIMENT NO: 1 (a)**

**AIM :** To implement program for FCFS scheduling Algorithm

**DESCRIPTION :** FCFS means First Come First Serve. That mean, based on arrival times of the processes into ready queue, CPU schedules the processes. By default, the procedure of FCFS shows that FCFS is a non – preemptive scheduling algorithm. Because, here there is no interruption while one process is executing. Here a process can execute until it’s burst time. Then only CPU schedules for another process.

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

wrt\_arrival\_time(x:list)

**PROGRAM:**

from texttable import Texttable

def wrt\_arrival\_time(x):

return x[1]

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of process : "))

l=[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of the process : ")

arrival = int(input("Enter the arival time of process in ms : "))

burst\_time =int(input("Enter the burst time in ms : "))

x = [name,arrival,burst\_time,0,0]

l.append(x)

l = sorted(l,key=wrt\_arrival\_time)

l[0][4]=l[0][2]

for i in range(1,n):

w=0

for j in range(0,i):

w+=l[j][2]

l[i][3]=w-l[i][1]

l[i][4]=l[i][2]+l[i][3]

total\_wt=0

total\_tt=0

for i in l:

total\_wt +=int(i[3])

total\_tt += int(i[4])

t = Texttable()

head = ['Process Name','Arrival Time','Burst Time','Wait Time','Turnaround Time']

l.insert(0,head)

t.add\_rows(l)

print(t.draw())

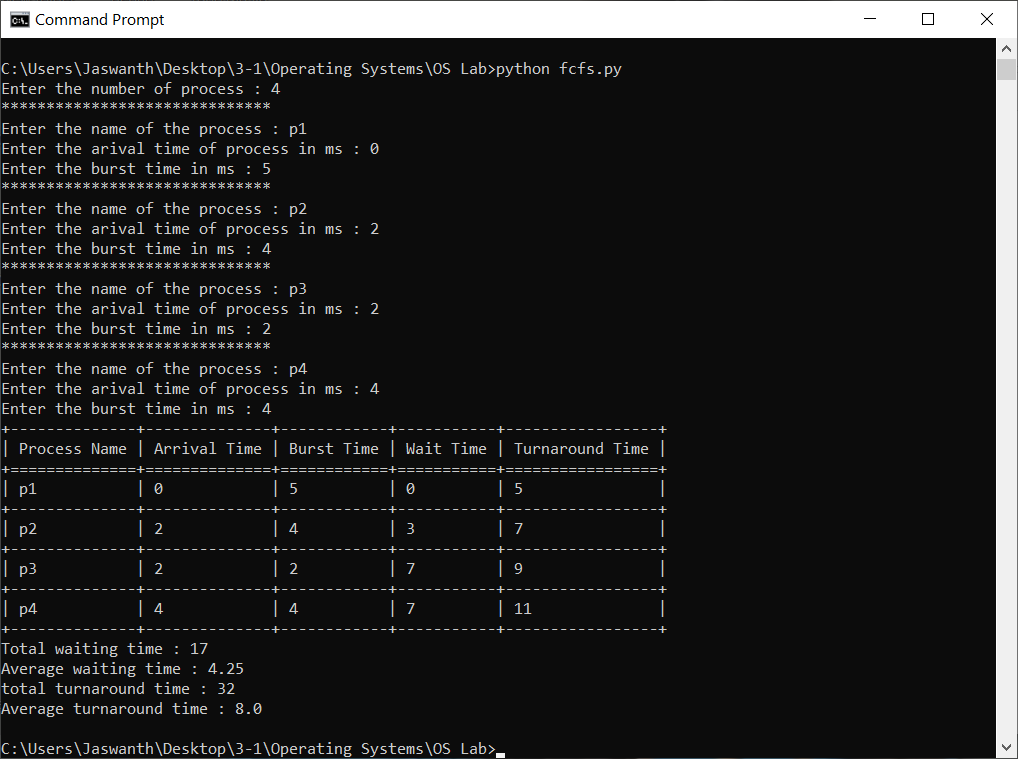
print("Total waiting time :",total\_wt)

print("Average waiting time :",total\_wt/n)

print("total turnaround time :",total\_tt)

print("Average turnaround time :",total\_tt/n)

**OUTPUT SCREEN SHOT:**



**EXPERIMENT NO: 1 (b)**

**AIM :** To implement program for Non Pre-emptive SJF scheduling Algorithm

**DESCRIPTION :** Shortest job first (SJF) or shortest job next, is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Shortest Job first has the advantage of having a minimum average waiting time among all scheduling algorithms.In this non-preemptive sjf,there is no interruption for the process in execution until its burst time.

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

class process:

\_\_init\_\_(sno,name,arrival,burst)

to\_list()

sjf(d:list)

wrt\_at(x:list)

wrt\_sno(x:list):

**PROGRAM:**

from texttable import Texttable

class process:

def \_\_init\_\_(self, sno, name, arrival, burst):

self.sno = sno

self.name = name

self.arrival = arrival

self.burst = burst

self.wt = 0

self.tt = 0

self.ct = 0

def to\_list(self):

return [self.sno,self.name,self.arrival,self.burst,self.wt,self.tt,self.ct]

def sjf(d):

t = Texttable()

t.add\_row(["S.No","Process name","Arrival time","Burst time","Wait time","Turnaorund time","Completion time"])

clock = 0

temp = []

l = []

total\_wt=0

total\_tt=0

n = len(d)

while len(d) > 0:

d= sorted(d,key=wrt\_at)

for at in d:

if at.arrival <= clock:

temp.append(at)

temp = sorted(temp,key=wrt\_bt)

if len(temp)==0:

clock+=1

continue

clock+=temp[0].burst

temp[0].ct=clock

temp[0].tt=temp[0].ct - temp[0].arrival

temp[0].wt=temp[0].tt- temp[0].burst

total\_tt+=temp[0].tt

total\_wt+=temp[0].wt

l.append(temp[0])

d.remove(temp[0])

temp.clear()

l = sorted(l,key = wrt\_sno)

for i in l:

t.add\_row(i.to\_list())

print(t.draw())

print("Total waiting time :",total\_wt)

print("Averge waiting time :",total\_wt/n)

print("Total turnaround time :",total\_tt)

print("Average turnaround time :",total\_tt/n)

def wrt\_at(x):

return x.arrival

def wrt\_bt(x):

return x.burst

def wrt\_sno(x):

return x.sno

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of processes : "))

d=[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of process :")

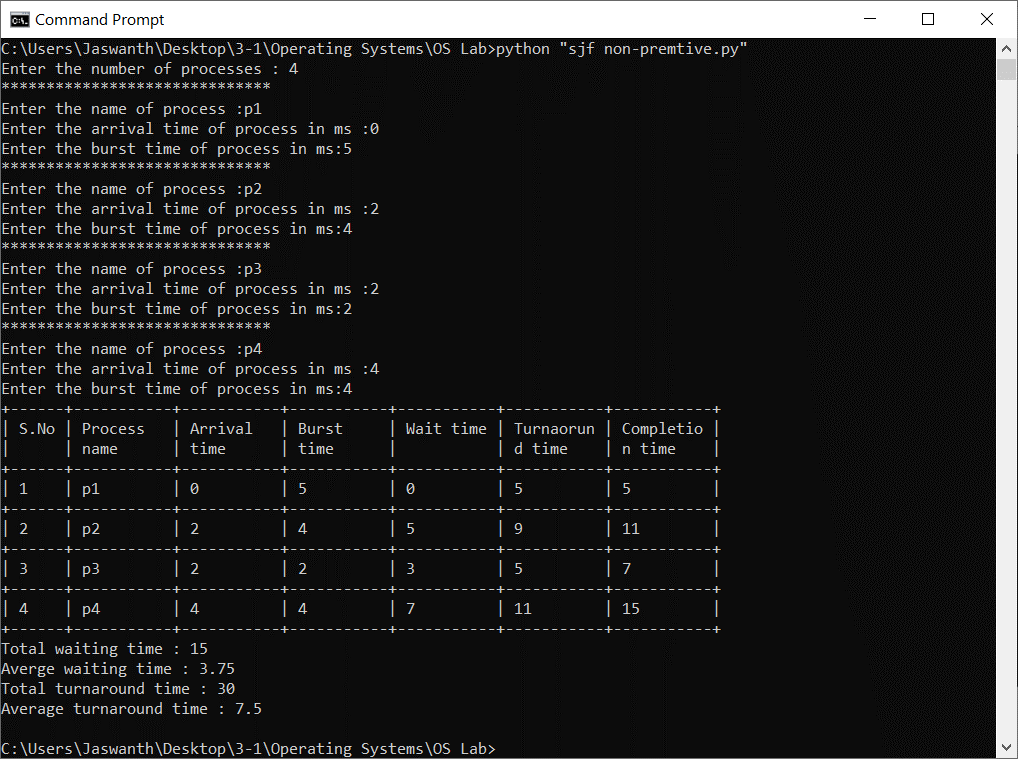
at = int(input("Enter the arrival time of process in ms :"))

bt = int(input("Enter the burst time of process in ms:"))

d.append(process(i+1,name,at,bt))

sjf(d)

**OUTPUT SCREENSHOT:**



**EXPERIMENT NO: 1 (c)**

**AIM :** To implement program for Preemptive sjf scheduling Algorithm

**DESCRIPTION :** SJF: Shortest Job First.

The task is to find the Average Waiting Time and Average Turnaround Time of the given processes with their Burst Time using SJF Scheduling Algorithm. SJF is a scheduling policy that selects the waiting process with the smallest execution time to execute next.

Priority Scheduling is a Non Pre-emptive and Pre-emptive Algorithm, hence the process which has the Least Burst Time is selected first. Here we are considering Pre-emptive version of Priority Scheduling, hence the process which has the Least Burst Time will be served first and will be continued to be served till there is any other process with Lower Burst Time priority. If there is any process with Lower Burst Time, then switch the process.

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

class Process:

\_\_init\_\_(sno,name,arrival,burst)

to\_list()->list[]

sjf(d:list)

wrt\_at(x:list)

wrt\_rem(x:list)

wrt\_sno(x:list)

**PROGRAM:**

def priority(d):

t = Texttable()

t.add\_row(["S.No","Process name","Arrival time","Burst time","Wait time","Turnaorund time","Completion time"])

clock = 0

temp = []

l = []

total\_wt=0

total\_tt=0

n = len(d)

while len(d) > 0:

d= sorted(d,key=wrt\_at)

for at in d:

if at.arrival <= clock:

temp.append(at)

temp = sorted(temp,key=wrt\_rem)

if len(temp)==0:

clock+=1

continue

clock+=1

temp[0].rem -=1

if temp[0].rem ==0:

temp[0].ct=clock

temp[0].tt=temp[0].ct - temp[0].arrival

temp[0].wt=temp[0].tt- temp[0].burst

total\_tt+=temp[0].tt

total\_wt+=temp[0].wt

l.append(temp[0])

d.remove(temp[0])

temp.clear()

l = sorted(l,key = wrt\_sno)

for i in l:

t.add\_row(i.to\_list())

print(t.draw())

print("Total waiting time :",total\_wt)

print("Averge waiting time :",total\_wt/n)

print("Total turnaround time :",total\_tt)

print("Average turnaround time :",total\_tt/n)

def wrt\_at(x):

return x.arrival

def wrt\_rem(x):

return x.rem

def wrt\_sno(x):

return x.sno

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of processes : "))

d=[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of process : ")

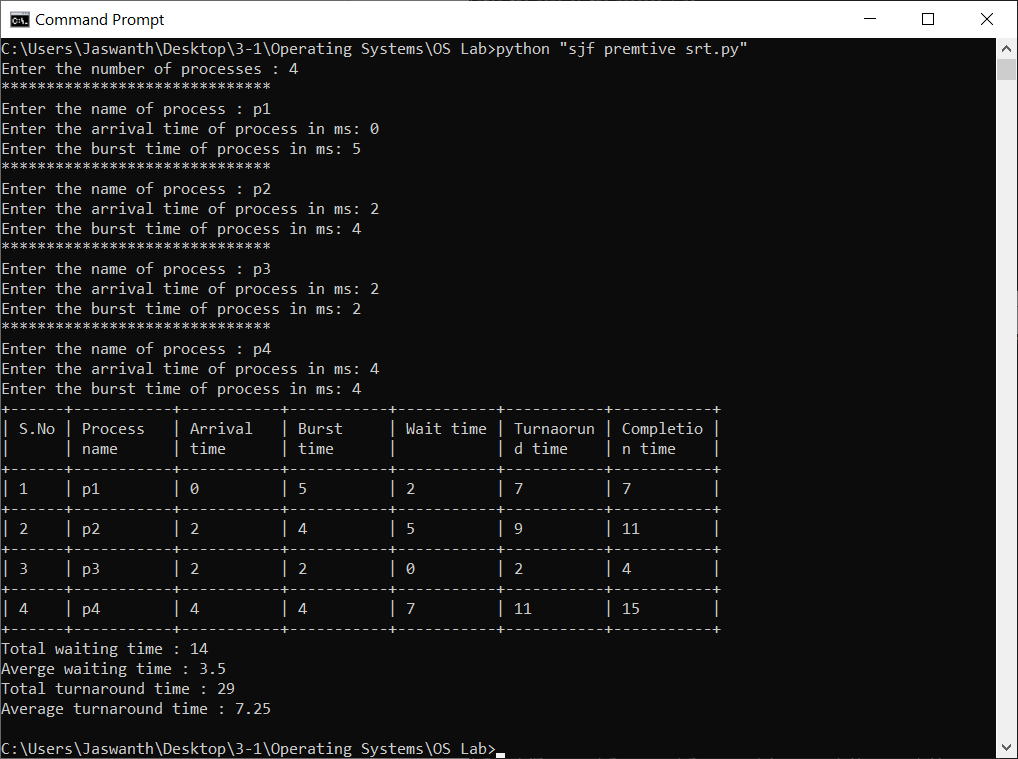
at = int(input("Enter the arrival time of process in ms: "))

bt = int(input("Enter the burst time of process in ms: "))

d.append(process(i+1,name,at,bt))

priority(d)

**OUTPUT SCREENSHOT:**



**EXPERIMENT NO: 1 (d)**

**AIM :** To implement program for Non-Preemptive Priority scheduling Algorithm

**DESCRIPTION :** The task is to find the Average Waiting Time and Average Turnaround Time of the given processes with their Burst Time using Priority Scheduling Algorithm.Priority is a scheduling policy that selects the waiting process with the highest priority to execute next.

Priority Scheduling is a Non Pre-emptive and Pre-emptive Algorithm, hence the process which has the Highest Priority is selected first.Here we are considering Non Pre-emptive version of Priority Scheduling, hence the process which has the Highest Priority will be served first and the next process will be served only after the previous process is executed completely.

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

class Process:

\_\_init\_\_(sno,name,arrival,burst,priority)

to\_list()->list[]

priority(d:list)

wrt\_at(x:list)

wrt\_p(x:list)

wrt\_sno(x:list)

**PROGRAM:**

from texttable import Texttable

class process:

def \_\_init\_\_(self, sno, name, arrival, burst,priority):

self.sno = sno

self.name = name

self.arrival = arrival

self.burst = burst

self.wt = 0

self.tt = 0

self.ct = 0

self.priority=priority

def to\_list(self):

return [self.sno,self.name,self.arrival,self.burst,self.priority,self.wt,self.tt,self.ct]

def priority(d):

t = Texttable()

t.add\_row(["S.No","Process name","Arrival time","Burst time","Priority","Wait time","Turnaorund time","Completion time"])

clock = 0

temp = []

l = []

total\_wt=0

total\_tt=0

n = len(d)

while len(d) > 0:

d= sorted(d,key=wrt\_at)

for at in d:

if at.arrival <= clock:

temp.append(at)

temp = sorted(temp,key=wrt\_p)

clock+=temp[0].burst

temp[0].ct=clock

temp[0].tt=temp[0].ct - temp[0].arrival

temp[0].wt=temp[0].tt-temp[0].burst

total\_tt+=temp[0].tt

total\_wt+=temp[0].wt

l.append(temp[0])

d.remove(temp[0])

temp.clear()

l = sorted(l,key = wrt\_sno)

for i in l:

t.add\_row(i.to\_list())

print(t.draw())

print("Total waiting time :",total\_wt)

print("Averge waiting time :",total\_wt/n)

print("Total turnaround time :",total\_tt)

print("Average turnaround time :",total\_tt/n)

def wrt\_at(x):

return x.arrival

def wrt\_p(x):

return x.priority

def wrt\_sno(x):

return x.sno

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of processes : "))

d=[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of process : ")

at = int(input("Enter the arrival time of the process: "))

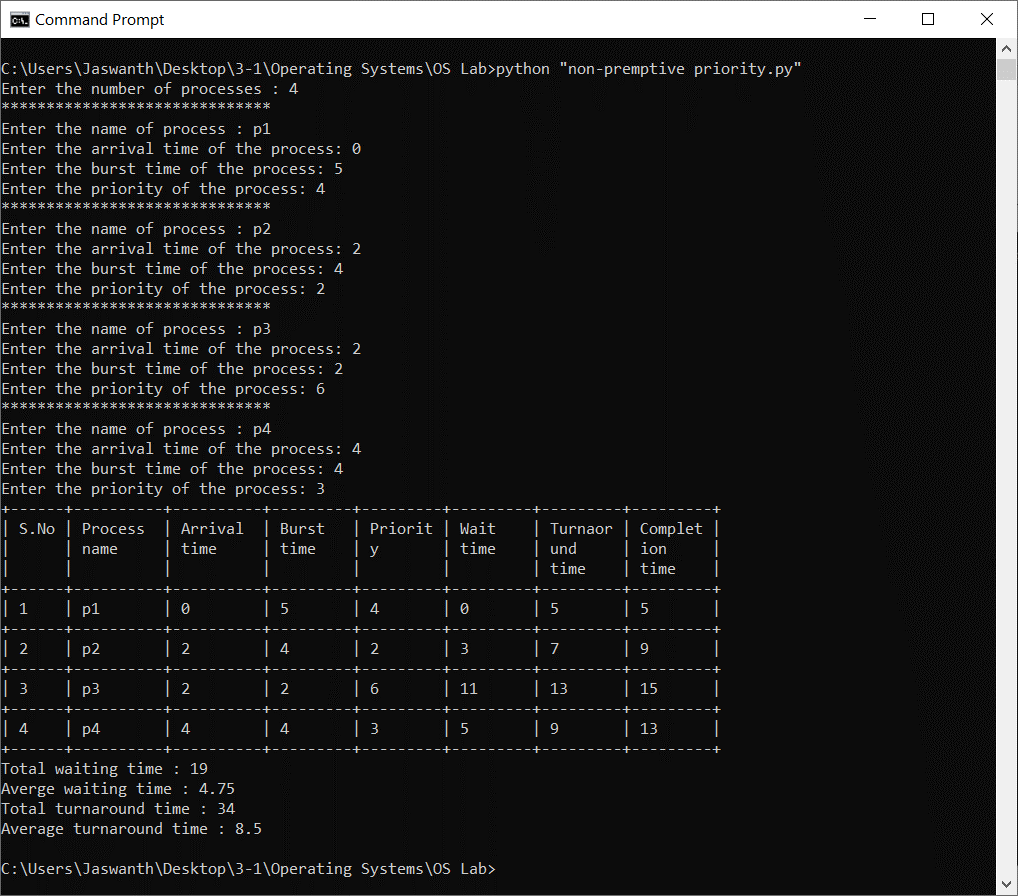
bt = int(input("Enter the burst time of the process: "))

p = int(input("Enter the priority of the process: "))

d.append(process(i+1,name,at,bt,p))

priority(d)

**OUTPUT SCREENSHOT:**



**EXPERIMENT NO: 1 (e)**

**AIM :** To implement program for Preemptive Priority scheduling Algorithm

**DESCRIPTION :** The task is to find the Average Waiting Time and Average Turnaround Time of the given processes with their Burst Time using Priority Scheduling Algorithm.

Priority is a scheduling policy that selects the waiting process with the highest priority to execute next.Priority Scheduling is a Non Pre-emptive and Pre-emptive Algorithm, hence the process which has the Highest Priority is selected first.

Here we are considering Pre-emptive version of Priority Scheduling, hence the process which has the Highest Priority will be served first and will be continued to be served till there is any other process with higher priority.If there is any process with higher priority, then switch the process.

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

class Process:

\_\_init\_\_(sno,name,arrival,burst,priority)

to\_list()->list[]

priority(d:list)

wrt\_at(x:list)

wrt\_p(x:list)

wrt\_sno(x:list)

**PROGRAM:**

from texttable import Texttable

class process:

def \_\_init\_\_(self, sno, name, arrival, burst,priority):

self.sno = sno

self.name = name

self.arrival = arrival

self.burst = burst

self.rem = burst

self.wt = 0

self.tt = 0

self.ct = 0

self.priority=priority

def to\_list(self):

return [self.sno,self.name,self.arrival,self.burst,self.priority,self.wt,self.tt,self.ct]

def priority(d):

t = Texttable()

t.add\_row(["S.No","Process name","Arrival time","Burst time","Priority","Wait time","Turnaorund time","Completion time"])

clock = 0

temp = []

l = []

total\_wt=0

total\_tt=0

n = len(d)

while len(d) > 0:

d= sorted(d,key=wrt\_at)

for at in d:

if at.arrival <= clock:

temp.append(at)

temp = sorted(temp,key=wrt\_p)

clock+=1

temp[0].rem-=1

if temp[0].rem==0:

temp[0].ct=clock

temp[0].tt=temp[0].ct - temp[0].arrival

temp[0].wt=temp[0].tt- temp[0].burst

total\_tt+=temp[0].tt

total\_wt+=temp[0].wt

l.append(temp[0])

d.remove(temp[0])

temp.clear()

l = sorted(l,key = wrt\_sno)

for i in l:

t.add\_row(i.to\_list())

print(t.draw())

print("Total waiting time :",total\_wt)

print("Averge waiting time :",(total\_wt/n))

print("Total turnaround time :",total\_tt)

print("Average turnaround time :",(total\_tt/n))

def wrt\_at(x):

return x.arrival

def wrt\_p(x):

return x.priority

def wrt\_sno(x):

return x.sno

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input("Enter the number of processes : "))

d=[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of process : ")

at = int(input("Enter the arrival time of the process: "))

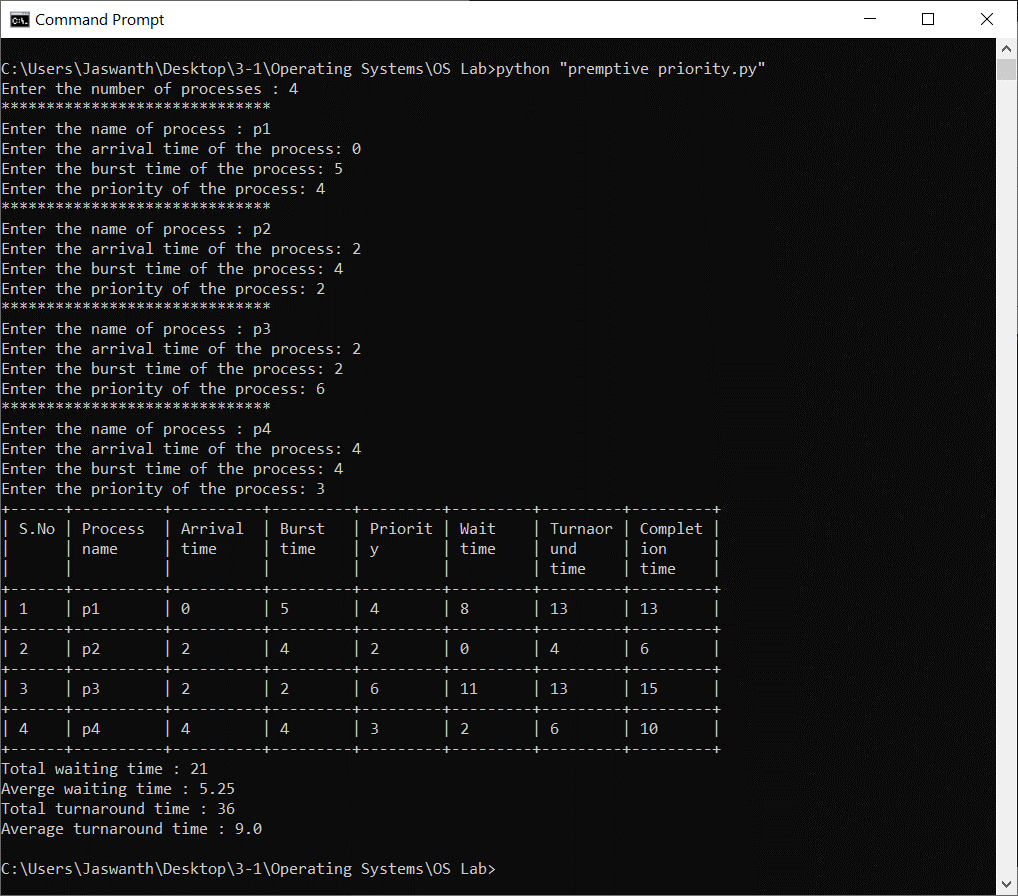
bt = int(input("Enter the burst time of the process: "))

p = int(input("Enter the priority of the process: "))

d.append(process(i+1,name,at,bt,p))

priority(d)

**OUTPUT SCREENSHOT:**



**EXPERIMENT NO: 1 (f)**

**AIM :** To implement program for Round Robin scheduling Algorithm

**DESCRIPTION :** The task is to find the Average Waiting Time and Average Turnaround Time of the given processes with their Burst Time using Round Robin Scheduling Algorithm.

Round Robin is a scheduling policy that selects the waiting process and executes it for a fixed time quantumRound Robin is a Pre-emptive Algorithm, hence the process will execute for a fixed time quantum and then it is switched and another process is executedRound Robin is cyclic in nature so it does not cause starvation.We will consider processes having different Arrival Time

**PROGRAMMING LANGUAGE USED:** PYTHON

**LIBRARIES USED:**  No built-in libraries used

**SYNTAX:**

class process:

\_\_init\_\_(sno,name,arrival,burst)

to\_list()

deb()

roundrobin(d:dictionary,quant:int)

wrt\_sno(x:list)

PROGRAM**:**

def roundrobin(d,quant):

t = Texttable()

t.add\_row(["S.No","Process name","Arrival time","Burst time","Wait time","Turnaround time"])

l = list(d.keys())

l.remove(0)

que = list()

q = []

clock = 0

i = 0

total\_wt=0

total\_tt=0

que.append(d[0])

while len(que)>0:

if que[i].rem>quant:

clock+=quant

que[i].rem -=quant

for at in l:

if at <= clock:

if isinstance(d[at],list):

que.extend(d[at])

else:

que.append(d[at])

l.remove(at)

else:

break

que.append(que[i])

que.remove(que[i])

elif que[i].rem == quant:

clock+=quant

que[i].rem = 0

que[i].wt = clock - que[i].burst-que[i].arrival

que[i].tt = que[i].wt + que[i].burst

total\_tt+=que[i].tt

total\_wt+=que[i].wt

q.append(que[i])

que.remove(que[i])

else:

clock+=que[i].rem

que[i].rem=0

que[i].wt = clock - que[i].burst-que[i].arrival

que[i].tt = que[i].wt + que[i].burst

total\_tt+=que[i].tt

total\_wt+=que[i].wt

q.append(que[i])

que.remove(que[i])

q = sorted(q,key = wrt\_sno)

for i in q:

t.add\_row(i.to\_list())

print(t.draw())

print("Total waiting time :",total\_wt)

print("Averge waiting time :",(total\_wt/len(d)))

print("Total turnaround time :",total\_tt)

print("Average turnaround time :",(total\_tt/len(d)))

def wrt\_sno(x):

return x.sno

if \_\_name\_\_ == "\_\_main\_\_":

q = int(input("Enter the quantum in ns : "))

n = int(input("Enter the number of processes : "))

d,l={},[]

for i in range(n):

print(30\*'\*')

name = input("Enter the name of process : ")

at = int(input("Enter the arrival time of the process in ms: "))

bt = int(input("Enter the burst of the process in ms : "))

if at not in d.keys():

d[at]=process(i+1,name,at,bt)

else:

d[at]=[d[at],process(i+1,name,at,bt)]

roundrobin(d,q)

**OUTPUT SCREENSHOT:**

