Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

WEEK-4

Program-1: Implement Priority CPU Scheduling algorithms without arrival times.

CODE:

```
def getWT(pid, pri, bt, n):
    p2 = pid[:]
    pri2 = pri[:]
    bt2 = bt[:]
    flag = False
    for i in range(n-1):
        for j in range(n-i-1):
            if pri2[j]>pri2[j+1]:
                flag = True
                p2[j], p2[j+1] = p2[j+1], p2[j]
                pri2[j], pri2[j+1] = pri2[j+1], pri2[j]
                bt2[j], bt2[j+1] = bt2[j+1], bt2[j]
        if flag == False:
            break
    wt = [[p2[0], 0]]
    #calculating Waiting time
    #i.e., Waiting Time = Entering Time - Arrival Time
    for i in range(1, n):
        wt.append([p2[i], wt[i-1][1] + bt2[i-1]])
    return p2, pri2, bt2, wt
def getTAT(p, pri, bt, n, wt):
    tat = []
    for i in range(n):
        #Turn Around Time = Waiting Time + Burst Time
        tat.append(wt[i][1] + bt[i])
```

```
    return tat
def getAvgTimes(pid, pri, bt, n):
    totalWT = totalTAT = 0
    p2, pri2, bt2, wt = getWT(pid, pri, bt, n)
    tat = getTAT(pid, pri, bt, n, wt)
    print("Priority CPU scheduling Algorithm(Without
AT):\n==========================================\n")
    print("PID\tPriority\tBT\tWT\tTAT\t")
    for i in range(n):
        print("{}\t{}\t\t{}\t{}\t{}\t".format(pid[i], pri[i], bt[i], wt[i][1], tat[i]))
    print()
    print("Processes Execution Sequence: ")
    print('p'+" -->p".join([str(i) for i in p2]))
    print()
    avgWT = sum([i[1] for i in wt])/n
    avgTAT = sum(tat)/n
    print("Average Waiting Time(WT): ",avgWT)
    print("Average Turn Around Time: ",avgTAT)
processes = [1,2,3,4]
priorities = [2,1,3,4]
bt = [4,5,6,3]
n = len(processes)
getAvgTimes(processes, priorities, bt, n)
```

Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

OUTPUT:

```
Priority CPU scheduling Algorithm(Without AT):
================================================

PID     Priority        BT      WT      TAT
1       2               4       0       4
2       1               5       5       10
3       3               6       9       15
4       4               3       15      18

Processes Execution Sequence:
p2 -->p1 -->p3 -->p4

Average Waiting Time(WT):  7.25
Average Turn Around Time:  11.75
```

Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

Program-2: Implement CPU scheduling algorithms with arrival times.

CODE:

```
d = {}
n = int(input("Enter the number of processes: "))
for i in range(n):
  p = "P" + str(i)
  x = int(input(f"Enter the burst time for {i}th process: "))
  y = int(input(f"Enter the arrival time for {i}th process: "))
  z = int(input(f"Enter the priority for {i}th process"))
  emp = []
  emp.append(y)
  emp.append(x)
  emp.append(z)
  d[p] = emp
d = dict(sorted(d.items(), key=lambda item: item[1][2]))
print(d)
curr , turn  = list(d.items())[0][1][0] , 0
wait_turn = {}
for i in d.keys():
  turn  = curr - d[i][0]
  wait_turn[i] = []
  wait_turn[i].append(turn)
  curr += d[i][1]
  wait_turn[i].append(curr - d[i][0])
  print(wait_turn)
avg_t , avg_w  = 0 , 0
l = len(wait_turn.values())
for i in wait_turn.values():
    avg_w += i[0]
    avg_t += i[1]
```

print(f"The average wait time is {avg_w / l} and the average turn around time is {avg_t / l}")

OUTPUT:

```
Enter the number of processes: 4
Enter the burst time for 0th process: 4
Enter the arrival time for 0th process: 0
Enter the priority for 0th process2
Enter the burst time for 1th process: 5
Enter the arrival time for 1th process: 1
Enter the priority for 1th process1
Enter the burst time for 2th process: 6
Enter the arrival time for 2th process: 0
Enter the priority for 2th process3
Enter the burst time for 3th process: 4
Enter the arrival time for 3th process: 2
Enter the priority for 3th process4
{'P1': [1, 5, 1], 'P0': [0, 4, 2], 'P2': [0, 6, 3], 'P3': [2, 4, 4]}
{'P1': [0, 5]}
{'P1': [0, 5], 'P0': [6, 10]}
{'P1': [0, 5], 'P0': [6, 10], 'P2': [10, 16]}
{'P1': [0, 5], 'P0': [6, 10], 'P2': [10, 16], 'P3': [14, 18]}
The average wait time is 7.5 and the average turn around time is 12.25
```

Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

Program-3:Implement Preempitive priority CPU scheduling algorithms with arrival times.

CODE:

```
totalprocess = 6
proc = []
for i in range(6):
    l = []
    for j in range(5):
        l.append(0)
    proc.append(l)
# Using FCFS Algorithm to find Waiting time
def get_wt_time( wt
    # declaring service array that stores
    # cumulative burst time
    service = [0] * 6
    # Initialising initial elements
    # of the arrays
    service[0] = 0
    wt[0] = 0
    for i in range(1, totalprocess):
        service[i] = proc[i - 1][1] + service[i - 1]
        wt[i] = service[i] - proc[i][0] + 1
        # If waiting time is negative,
        # change it o zero
        if(wt[i] < 0) :
            wt[i] = 0
def get_tat_time(tat, wt):
    # Filling turnaroundtime array
    for i in range(totalprocess):
        tat[i] = proc[i][1] + wt[i]
def findgc():
```

```
# Declare waiting time and
# turnaround time array
wt = [0] * 6
tat = [0] * 6
wavg = 0
tavg = 0
# Function call to find waiting time array
get_wt_time(wt)
# Function call to find turnaround time
get_tat_time(tat, wt)
stime = [0] * 6
ctime = [0] * 6
stime[0] = 1
ctime[0] = stime[0] + tat[0]
# calculating starting and ending time
for i in range(1, totalprocess):
    stime[i] = ctime[i - 1]
    ctime[i] = stime[i] + tat[i] - wt[i]
print("Process_no\tStart_time\tComplete_time",
          "\tTurn_Around_Time\tWaiting_Time")
# display the process details
for i in range(totalprocess):
    wavg += wt[i]
    tavg += tat[i]
    print(proc[i][3], "\t\t", stime[i],
                  "\t\t", end = " ")
    print(ctime[i], "\t\t", tat[i], "\t\t\t", wt[i])
# display the average waiting time
# and average turn around time
print("Average waiting time is : ", end = " ")
```

```
    print(wavg / totalprocess)
    print("average turnaround time : " , end = " ")
    print(tavg / totalprocess)
# Driver code
if __name__ =="__main__":
    arrivaltime = [0,1,2,3,4,5,6]
    bursttime = [1,7,3,6,5,15,8]
    priority = [2,6,3,5,4,10,9]
    for i in range(totalprocess):
        proc[i][0] = arrivaltime[i]
        proc[i][1] = bursttime[i]
        proc[i][2] = priority[i]
        proc[i][3] = i + 1
    # Using inbuilt sort function
    proc = sorted (proc, key = lambda x:x[2])
    proc = sorted (proc)
    # Calling function findgc for
    # finding Gantt Chart
    findgc()
```

OUTPUT:

```
Process_no      Start_time      Complete_time   Turn_Around_Time        Waiting_Time
1                1              2               1                       0
2                2              9               8                       1
3                9              12              10                      7
4                12             18              15                      9
5                18             23              19                      14
6                23             38              33                      18
Average waiting time is :  8.166666666666666
average turnaround time :  14.333333333333334
```

Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

Program-4:Implement Round Robin CPU scheduling algorithms with arrival times.

CODE:

```
if __name__ == '__main__':
    # Python program for implementation of RR Scheduling
    print("Enter Total Process Number: ")
    total_p_no = int(input())
    total_time = 0
    total_time_counted = 0
    # proc is process list
    proc = []
    wait_time = 0
    turnaround_time = 0
    print("Enter process arrival time and burst time")
    for _ in range(total_p_no):
        # Getting the input for process
        input_info = list(map(int, input().split(" ")))
        arrival, burst, remaining_time = input_info[0], input_info[1], input_info[1]
        # processes are appended to the proc list in following format
        proc.append([arrival, burst, remaining_time, 0])
        # total_time gets incremented with burst time of each process
        total_time += burst
    print("Enter time quantum")
    time_quantum = int(input())
    # Keep traversing in round robin manner until the total_time == 0
    while total_time != 0:
        # traverse all the processes
        for i in range(len(proc)):
            # proc[i][2] here refers to remaining_time for each process i.e "i"
            if proc[i][2] <= time_quantum and proc[i][2] >= 0:
                total_time_counted += proc[i][2]
```

```
            total_time -= proc[i][2]
            # the process has completely ended here thus setting it's remaining time to 0.
            proc[i][2] = 0
        elif proc[i][2] > 0:
            # if process has not finished, decrementing it's remaining time by time_quantum
            proc[i][2] -= time_quantum
            total_time -= time_quantum
            total_time_counted += time_quantum
        if proc[i][2] == 0 and proc[i][3] != 1:
            # if remaining time of process is 0
            # and
            # individual waiting time of process has not been calculated i.e flag
            wait_time += total_time_counted - proc[i][0] - proc[i][1]
            turnaround_time += total_time_counted - proc[i][0]
            # flag is set to 1 once wait time is calculated
            proc[i][3] = 1
    print("\nAvg Waiting Time is ", (wait_time * 1) / total_p_no)
    print("Avg Turnaround Time is ", (turnaround_time * 1) / total_p_no)
```

Name: Mohammad Shireen **Roll no.: 160120748303** **Section; C4**

OUTPUT:

```
Enter Total Process Number:
6
Enter process arrival time and burst time
0 25
25 25
30 25
60 15
100 10
10 105
Enter time quantum
20

Avg Waiting Time is  46.666666666666664
Avg Turnaround Time is  80.83333333333333
```