# EXPERIMENT NO-3

**Aim: -** To implement Shortest Job First - SJF process scheduling algorithm

**Lab objective:** Describe Process & process management using CPU scheduling Algorithm

# Theory:-

In this algorithm the process with the shorted burst time is allocated to CPU first. If the first process arrives then the CPU is allocated to it. Meanwhile if the 2nd process arrives, the burst time of this process is compared with the remaining burst time of the 1st process. If the burst time of just arrived process is less than that existing process the execution of current process is stopped and the execution of 2nd process begins. On the reverse cane, the execution of current process is continued and the 2nd process goes to waiting queue. This algorithm is appropriate for batch jobs for which run times are known in advanced. SJF scheduling algorithm gives minimum average waiting time for given set of process. It requires precise knowledge of how long a job or process will run.

# Code:-

def findWaitingTime(processes, n, wt): rt = [0] \* n

for i in range(n):

rt[i] = processes[i][1] complete = 0

t = 0

minm = 999999999

short = 0 check = False

while complete != n: for j in range(n):

if ((processes[j][2] <= t) and (rt[j] < minm) and rt[j] > 0):

minm = rt[j] short = j check = True

if check == False: t += 1

continue

rt[short] -= 1

minm = rt[short] if minm == 0:

minm = 999999999

if rt[short] == 0:

complete += 1 check = False

fint = t + 1

wt[short] = (fint - proc[short][1] - proc[short][2]) if wt[short] < 0:

wt[short] = 0

t += 1

def findTurnAroundTime(processes, n, wt, tat): for i in range(n):

tat[i] = processes[i][1] + wt[i]

def findavgTime(processes, n): wt = [0] \* n

tat = [0] \* n findWaitingTime(processes, n, wt)

findTurnAroundTime(processes, n, wt, tat)

print("Processes Burst Time Waiting", "Time Turn- Around Time")

total\_wt = 0

total\_tat = 0

for i in range(n):

total\_wt = total\_wt + wt[i] total\_tat = total\_tat + tat[i]

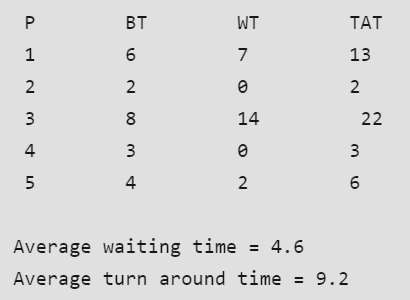
print(" ", processes[i][0], "\t\t", processes[i][1], "\t\t", wt[i], "\t\t", tat[i])

print("\nAverage waiting time = %.5f " % (total\_wt / n)) print("Average turn around time = ", total\_tat / n)

proc = [[1, 6, 1], [2, 8, 1], [3, 7, 2], [4, 3, 3]]

n = 4 findavgTime(proc, n)

# Output:



**Discussion:** Shortest Job First is a CPU Scheduling Algorithm which focuses on the burst time of the processes. The process having the shortest burst time is executed first. In the above example there are four processes namely P1,P2,P3 and P4. Their Burst Time is 6,8,7,3 respectively. The code finds the shortest execution time of each process at the current time and that process is given priority while the other process is in the ready queue until the first process completes its execution. Likewise the burst time is checked at every point and all the processes are completed. This how the Preemptive Shortest Job First Algorithm works.

# Lab Outcome:-

CPU Scheduling is a process of determining which process will own CPU for execution while another process is on hold. The successful implementation of SJF algorithm helps to understand that the process with the shorted burst time is allocated to CPU first**.**

# Conclusion:-

* Shortest job first is a scheduling algorithm in which the process with the shortest execution time should be selected for execution next.
* This is the best approach to minimize waiting time.