**Experiment No: 05**

**Aim:** Write a program to demonstrate the concept of preemptive scheduling algorithms & non-preemptive scheduling algorithms.

**Theory:**

**1. Preemptive Scheduling:**

Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.

Algorithms based on preemptive scheduling are: [Round Robin (RR)](https://www.geeksforgeeks.org/program-round-robin-scheduling-set-1/),[Shortest Remaining Time First (SRTF)](https://www.geeksforgeeks.org/program-shortest-job-first-scheduling-set-2srtf-make-changesdoneplease-review/), [Priority (preemptive version)](https://www.geeksforgeeks.org/program-for-preemptive-priority-cpu-scheduling/), etc.

**2. Non-Preemptive Scheduling:**

Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.

Algorithms based on non-preemptive scheduling are:[Shortest Job First (SJF basically non preemptive)](https://www.geeksforgeeks.org/program-shortest-job-first-sjf-scheduling-set-1-non-preemptive/) and [Priority (non preemptive version)](https://www.geeksforgeeks.org/operating-system-priority-scheduling-different-arrival-time-set-2/), etc.

**Program:**

#include<stdio.h>

int main()

{

int n, bt[20], wt[20], tat[20], avwt = 0, avtat = 0, i, j;

printf("Enter total number of processes(maximum 20):");

scanf("%d", &n);

printf("\nEnter Process Burst Time\n");

for (i = 0; i < n; i++)

{

printf("P[%d]:", i + 1);

scanf("%d", &bt[i]);

}

wt[0] = 0;

for (i = 1; i < n; i++)

{

wt[i] = 0;

for (j = 0; j < i; j++)

wt[i] += bt[j];

}

printf("\nProcess\t\tBurst Time\tWaiting Time\tTurnaround Time");

for (i = 0; i < n; i++)

{

tat[i] = bt[i] + wt[i];

avwt += wt[i];

avtat += tat[i];

printf("\nP[%d]\t\t%d\t\t%d\t\t%d", i + 1, bt[i], wt[i], tat[i]);

}

avwt /= i;

avtat /= i;

printf("\n\nAverage Waiting Time:%d", avwt);

printf("\nAverage Turnaround Time:%d", avtat);

return 0;

}

**Output:**

Enter total number of processes(maximum 20):5

Enter Process Burst Time

P[1]:5

P[2]:2

P[3]:1

P[4]:4

P[5]:3

Process Burst Time Waiting Time Turnaround Time

P[1] 5 0 5

P[2] 2 5 7

P[3] 1 7 8

P[4] 4 8 12

P[5] 3 12 15

Average Waiting Time:6

Average Turnaround Time:9

**Conclusion:** We have successfully wrote a program to demonstrate the concept of preemptive scheduling algorithms & non-preemptive scheduling algorithms.