Department of Computer Engineering

Batch: E-2	**Roll No.: 16010123325**
Experiment No. 05	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Implementation of Priority Based CPU Scheduling Algorithms – Pre-emptive and Non Pre-emptive.

AIM: Implementation of Priority Based CPU Scheduling Algorithms – Pre-emptive Non Preemptive

Expected Outcome of Experiment:

CO 2 Illustrate and analyse the Process, threads, process scheduling and thread scheduling

Books/ Journals/ Websites referred:

1. **Silberschatz A., Galvin P., Gagne G. "Operating Systems Principles", Willey Eight edition.**
2. **Achyut S. Godbole , Atul Kahate "Operating Systems" McGraw Hill Third Edition.**
3. **William Stallings, "Operating System Internal & Design Principles", Pearson.**
4. **Andrew S. Tanenbaum, "Modern Operating System", Prentice Hall.**

Pre Lab/ Prior Concepts:

Priority scheduling in OS is the scheduling algorithm that schedules processes according to the priority assigned to each of the processes. Higher priority processes are executed before lower priority processes.

What is a Priority Scheduling Algorithm?

The Operating System has a major function of deciding the order in which processes/tasks will have access to the processor and the amount of processing time each of these processes or tasks will have. This function of the operating system is termed process scheduling.

Under process scheduling, the tasks that the operating system performs are:

- Keeping track of the status of the processes
- Allocation of processor to processes
- De-allocation of the processor to processes.

All these processes are stored by the operating system in a process table along with an ID allotted to every process (PID), to help keep track of all of them. And, to keep track of their current state the operating system uses a Process Control Block (PCB). The information is updated in the control block as and when the state of the process changes.

But did you wonder what is the criteria that the Operating System uses for the scheduling of these processes? Which process must be allocated with CPU resources first?

Well, there are multiple ways in which the Operating System makes these decisions such as execution of the process that takes the least time first, or execution according to the order in which they requested access to the processor, and so on. One such important criterion is the priority of the process.

In priority scheduling in OS, processes are executed on the basis of their priority. The jobs/processes with higher priority are executed first. Naturally, you might want to know how the priority of processes is decided.

Priority of processes depends on some factors such as:

Time limit

Memory requirements of the process

Ratio of average I/O to average CPU burst time

There can be more factors on the basis of which the priority of a process/job is determined. This priority is assigned to the processes by the scheduler.

These priorities of processes are represented as simple integers in a fixed range such as 0 to 7, or maybe 0 to 4095. These numbers depend on the type of system.

Now that we know that each job is assigned a priority on the basis of which it is scheduled, we can move ahead with its working and types.

Types of Priority Scheduling Algorithms

There are two types of priority scheduling algorithms in OS:

Non-Preemptive Scheduling

In this type of scheduling:

If during the execution of a process, another process with a higher priority arrives for execution, even then the currently executing process will not be disturbed.

The newly arrived high priority process will be put in next for execution since it has higher priority than the processes that are in a waiting state for execution.

All the other processes will remain in the waiting queue to be processed. Once the execution of the current process is done, the high-priority process will be given the CPU for execution.

Preemptive Scheduling

`Preemptive Scheduling as opposed to non-preemptive scheduling will preempt (stop and store the currently executing process) the currently running process if a higher priority process enters the waiting state for execution and will execute the higher priority process first and then resume executing the previous process.

Characteristics of Priority Scheduling Algorithm

It is a scheduling algorithm that schedules the incoming processes on the basis of priority.

Operating systems use it for performing batch processes

If there exist two jobs/processes in the ready state (ready for execution) that have the same priority, then priority scheduling executes the processes on a first come first serve basis. For every job that exists, we have a priority number assigned to it that indicates its priority level.

If the integer value of the priority number is low, it means that the process has a higher priority. (low number = high priority).

Description of the application to be implemented:

Pre-emptive :

In this approach, the CPU allocates time to processes based on their priority. If a new process with a higher priority arrives, it **preempts** the currently running process. This ensures that **higher-priority tasks execute first**, but it increases context switching.

Non Pre-emptive :

Here, the CPU schedules processes based on priority, but once a process starts execution, it runs to completion before switching to another process. Lower priority processes may face starvation if higher-priority ones keep arriving.

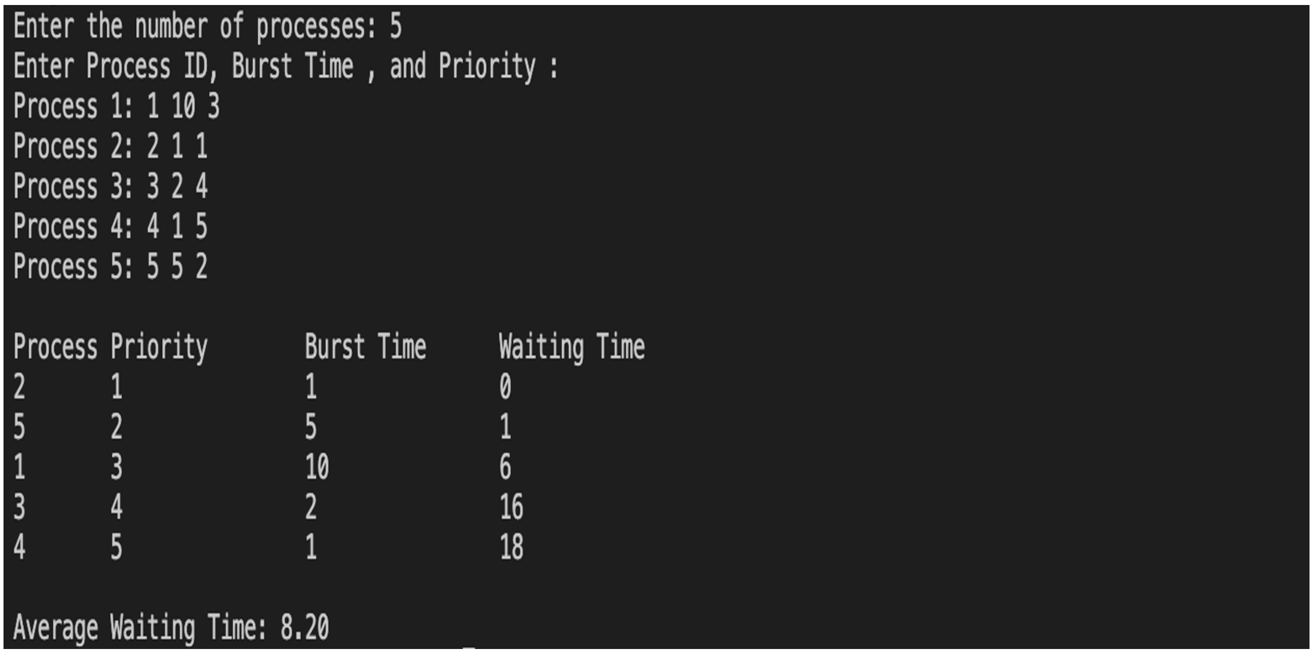
Implementation details:

Non Preemptive

```
#include <stdio.h>
#include <limits.h>

typedef struct Process {
    int pid;
    int bt;
    int priority;
}process;

void sort(process proc[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (proc[j].priority > proc[j + 1].priority) {
                process temp = proc[j];
                proc[j] = proc[j + 1];
                proc[j + 1] = temp;
            }
        }
    }
}

void findwt(process proc[], int n, int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + proc[i - 1].bt;
    }
}


void printResults(process proc[], int n) {
    int wt[n], tat[n], total_wt = 0;

    findwt(proc, n, wt);

    printf("\nProcess\tPriority\tBurst Time\tWaiting Time\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\t\t\n", proc[i].pid, proc[i].priority, proc[i].bt, wt[i]);
        total_wt += wt[i];
    }

    printf("\nAverage Waiting Time: %.2f\n", (float)total_wt / n);
}
```

```c
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    process proc[n];

    printf("Enter Process ID, Burst Time , and Priority :\n");
    for (int i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d %d %d", &proc[i].pid, &proc[i].bt, &proc[i].priority);
    }
    sort(proc,  n);
    printResults(proc, n);

}
```

Output-



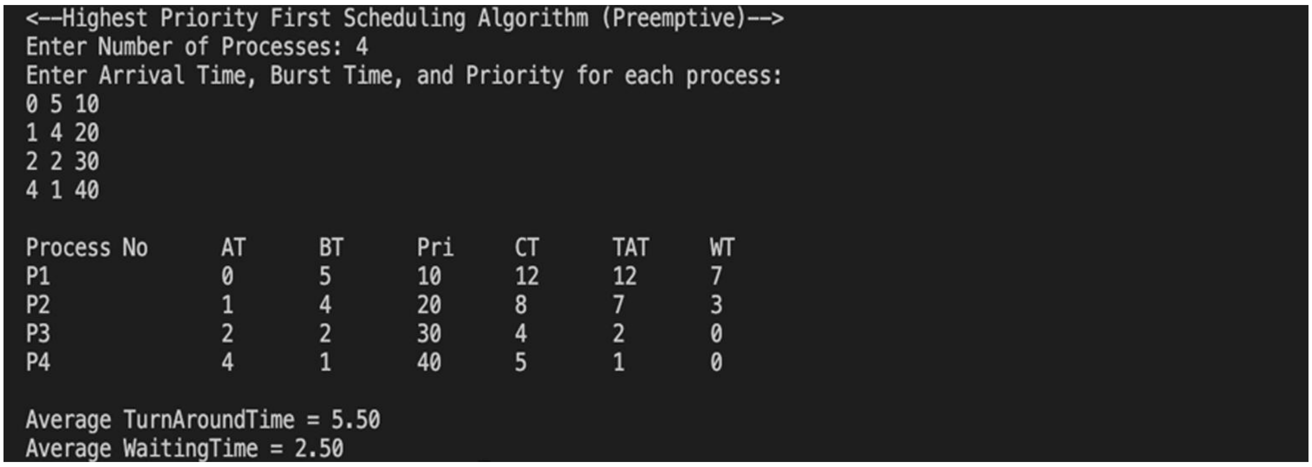
```
Enter the number of processes: 5
Enter Process ID, Burst Time , and Priority :
Process 1: 1 10 3
Process 2: 2 1 1
Process 3: 3 2 4
Process 4: 4 1 5
Process 5: 5 5 2

Process Priority        Burst Time      Waiting Time
2       1               1               0
5       2               5               1
1       3               10              6
3       4               2               16
4       5               1               18

Average Waiting Time: 8.20
```

Preemptive

```
#include <stdio.h>

#define MIN -9999

struct proc {
    int no, at, bt, rt, ct, wt, tat, pri, temp;
};


struct proc read(int i) {
    struct proc p;
    p.no = i;
    scanf("%d %d %d", &p.at, &p.bt, &p.pri);
    p.rt = p.bt;
    p.temp = p.pri;
    return p;
}

int main() {
    int i, n, c, remaining, max_val, max_index;
    struct proc p[10], temp;
    float avgtat = 0, avgwt = 0;

    printf("<--Highest Priority First Scheduling Algorithm (Preemptive)-->\n");
    printf("Enter Number of Processes: ");
    scanf("%d", &n);

    printf("Enter Arrival Time, Burst Time, and Priority for each process:\n");
    for (i = 0; i < n; i++)
        p[i] = read(i + 1);

    remaining = n;


    for (i = 0; i < n - 1; i++)
        for (int j = 0; j < n - i - 1; j++)
            if (p[j].at > p[j + 1].at) {
                temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }

    max_val = p[0].temp, max_index = 0;
    for (int j = 0; j < n && p[j].at <= p[0].at; j++)
```

```
        if (p[j].temp > max_val)
            max_val = p[j].temp, max_index = j;

    i = max_index;
    c = p[i].ct = p[i].at + 1;
    p[i].rt--;

    if (p[i].rt == 0) {
        p[i].temp = MIN;
        remaining--;
    }

    while (remaining > 0) {
        max_val = p[0].temp, max_index = 0;
        for (int j = 0; j < n && p[j].at <= c; j++)
            if (p[j].temp > max_val)
                max_val = p[j].temp, max_index = j;

        i = max_index;
        p[i].ct = c = c + 1;
        p[i].rt--;

        if (p[i].rt == 0) {
            p[i].temp = MIN;
            remaining--;
        }
    }

    printf("\nProcess No\tAT\tBT\tPri\tCT\tTAT\tWT\n");
    for (i = 0; i < n; i++) {
        p[i].tat = p[i].ct - p[i].at;
        avgtat += p[i].tat;
        p[i].wt = p[i].tat - p[i].bt;
        avgwt += p[i].wt;
        printf("P%d\t\t%d\t%d\t%d\t%d\t%d\t%d\n",
               p[i].no, p[i].at, p[i].bt, p[i].pri, p[i].ct, p[i].tat, p[i].wt);
    }

    avgtat /= n, avgwt /= n;
    printf("\nAverage TurnAroundTime = %.2f\nAverage WaitingTime = %.2f\n",
avgtat, avgwt);
}
```

Output-



```
<--Highest Priority First Scheduling Algorithm (Preemptive)-->
Enter Number of Processes: 4
Enter Arrival Time, Burst Time, and Priority for each process:
0 5 10
1 4 20
2 2 30
4 1 40

Process No      AT      BT      Pri     CT      TAT     WT
P1              0       5       10      12      12      7
P2              1       4       20      8       7       3
P3              2       2       30      4       2       0
P4              4       1       40      5       1       0

Average TurnAroundTime = 5.50
Average WaitingTime = 2.50
```

Conclusion :

The above experiment demonstrates implementation of Priority scheduling in both non-preemptive and non-preemptive methods.

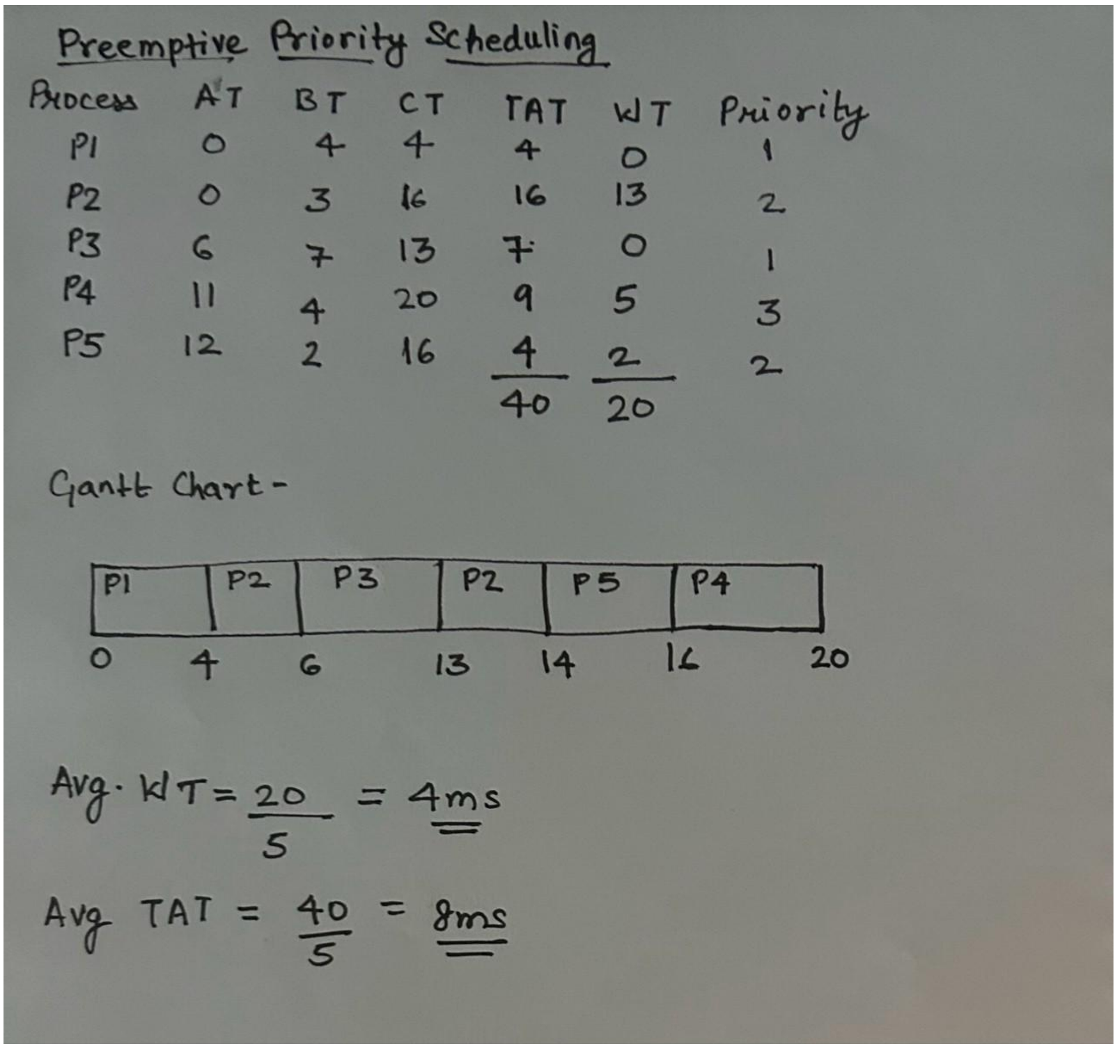
Post Lab Descriptive Questions

1. Consider a set of processes with their respective arrival and burst times given in milliseconds.

Process	Priority	Arrival Time	Burst Time
P1	1	0	4
P2	2	0	3
P3	1	6	7
P4	3	11	4
P5	2	12	2

A. **Apply the Priority based CPU scheduling algorithm.**

- Draw the Gantt Chart.
- Calculate the Average Turnaround Time.
- Calculate the Average Waiting Time.



Preemptive Priority Scheduling

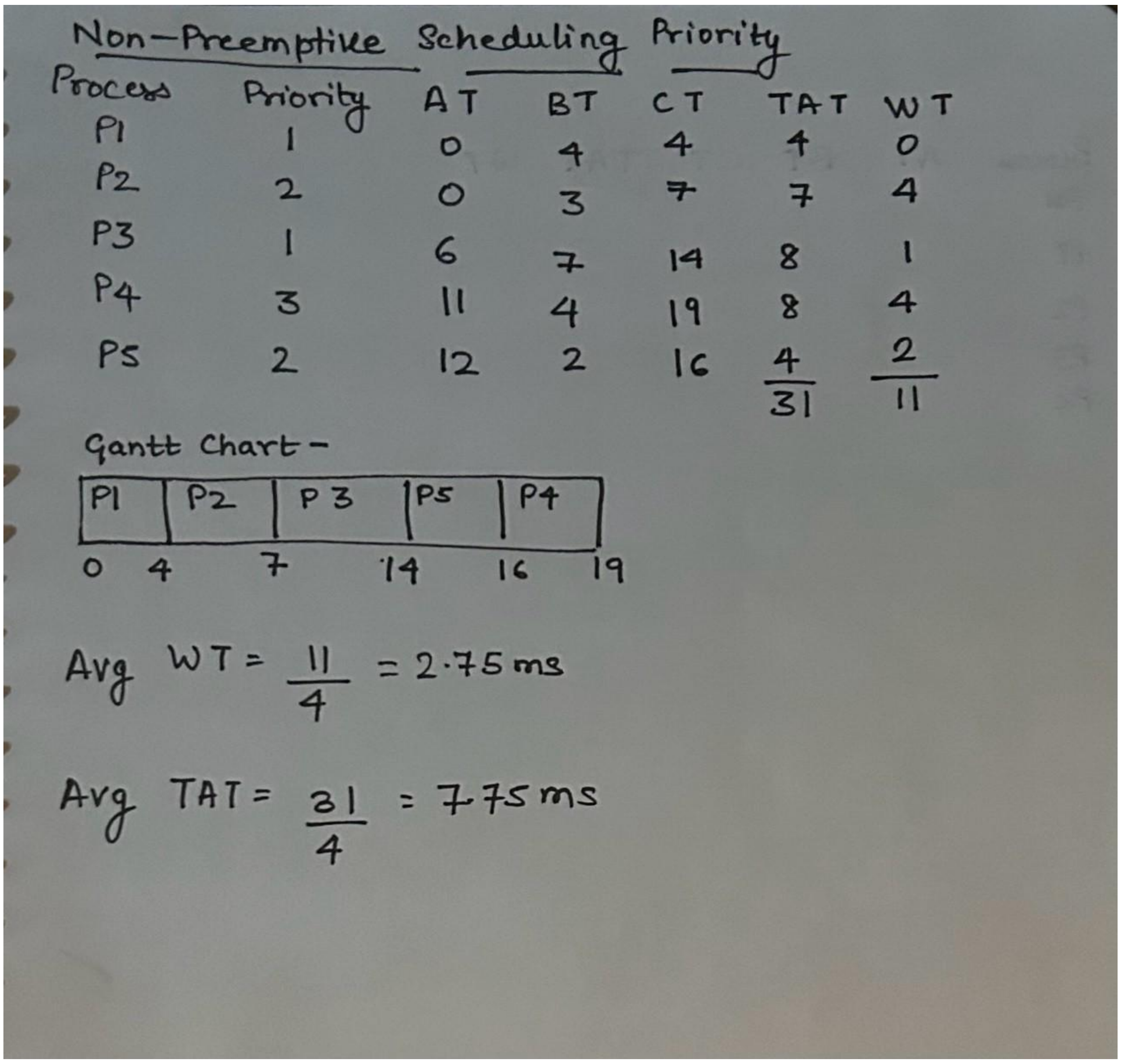
Process	AT	BT	CT	TAT	WT	Priority
P1	0	4	4	4	0	1
P2	0	3	16	16	13	2
P3	6	7	13	7	0	1
P4	11	4	20	9	5	3
P5	12	2	16	4	2	2
				40	20	

Gantt Chart -

```
| P1 | P2 | P3      | P2 | P5    | P4      |
0    4    6         13   14      16        20
```

$$\text{Avg. WT} = \frac{20}{5} = 4\text{ms}$$

$$\text{Avg TAT} = \frac{40}{5} = 8\text{ms}$$



Non-Preemptive Scheduling Priority

Process	Priority	AT	BT	CT	TAT	WT
P1	1	0	4	4	4	0
P2	2	0	3	7	7	4
P3	1	6	7	14	8	1
P4	3	11	4	19	8	4
P5	2	12	2	16	4	2
					31	11

Gantt Chart -

	P1	P2	P3	P5	P4
0	4	7	14	16	19

Avg WT = $\frac{11}{4}$ = 2.75 ms

Avg TAT = $\frac{31}{4}$ = 7.75 ms

B. Analyze the results and comment on the performance of pre-emptive and non-pre-emptive for the given process set.

Non-Preemptive Scheduling results in **lower waiting time** and **faster completion** for processes because once a process starts, it runs until completion.

Preemptive Scheduling results in **higher waiting time** and **higher turnaround time** due to frequent interruptions and context switching.

Date: ______________ **Signature of faculty in-charge**