

PRATICAL-5

AIM: List of processes / jobs along with their arrival times, priority value & CPU burst times is given.
Write a program to print the total waiting time, average waiting time, total turnaround time, average turnaround time & Gantt chart using Priority CPU scheduling policy.

● SOURCE CODE :

```
#include <stdio.h>
#include <stdbool.h>
#include <limits.h> // For INT_MAX

struct Process {
    int id, AT, BT, WT, TAT, CT, priority;
    bool isCompleted;
};

void priorityScheduling(struct Process p[], int n) {
    int total_WT = 0, total_TAT = 0, completed = 0, current_time = 0;
    int gantt_sequence[n], gantt_index = 0; // To store the order of execution

    while (completed < n) {
        int highest_priority_index = -1;
        int min_priority = INT_MAX;

        for (int i = 0; i < n; i++) {
            if (!p[i].isCompleted && p[i].AT <= current_time) {
                if (p[i].priority < min_priority || (p[i].priority == min_priority && p[i].AT < p[highest_priority_index].AT)) {
                    min_priority = p[i].priority;
                    highest_priority_index = i;
                }
            }
        }

        if (highest_priority_index == -1) {
            current_time++;
            continue;
        }

        int i = highest_priority_index;
        current_time = (current_time < p[i].AT) ? p[i].AT : current_time;
        p[i].CT = current_time + p[i].BT;
        p[i].TAT = p[i].CT - p[i].AT;
        p[i].WT = p[i].TAT - p[i].BT;

        total_WT += p[i].WT;
        total_TAT += p[i].TAT;
    }
}
```

```
current_time = p[i].CT;
p[i].isCompleted = true;
gantt_sequence[gantt_index++] = i; // Record the order of execution
completed++;
}
printf("\nProcess AT BT Priority WT TAT\n");
for (int i = 0; i < n; i++) {
    printf(" P%d %d %d %d %d\n", p[i].id, p[i].AT, p[i].BT, p[i].priority, p[i].WT, p[i].TAT);
    printf("\nTotal WT: %d\n", total_WT);
    printf("Average WT: %.2f\n", (float)total_WT / n);
    printf("Total TAT: %d\n", total_TAT);
    printf("Average TAT: %.2f\n", (float)total_TAT / n);
// Gantt Chart
printf("\nGantt Chart:\n");
for (int i = 0; i < gantt_index; i++) {
    printf(" P%d |", p[gantt_sequence[i]].id);
}
printf("\n0");
current_time = 0;
for (int i = 0; i < gantt_index; i++) {
    current_time = (current_time < p[gantt_sequence[i]].AT) ? p[gantt_sequence[i]].AT : current_time;
    current_time += p[gantt_sequence[i]].BT;
    printf(" %d", current_time);
}
printf("\n");
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process p[n];
    for (int i = 0; i < n; i++) {
        p[i].id = i + 1;
        p[i].isCompleted = false;
        printf("Enter AT, BT, and Priority for process P%d: ", p[i].id);
        scanf("%d %d %d", &p[i].AT, &p[i].BT, &p[i].priority);
    }
    priorityScheduling(p, n);
    return 0;
}
```

Output

```
Enter the number of processes: 4
Enter AT, BT, and Priority for process P1: 2 3 4
Enter AT, BT, and Priority for process P2: 1 5 3
Enter AT, BT, and Priority for process P3: 0 4 1
Enter AT, BT, and Priority for process P4: 3 6 2
```

Process	AT	BT	Priority	WT	TAT
P1	2	3	4	13	16
P2	1	5	3	9	14
P3	0	4	1	0	4
P4	3	6	2	1	7

Total WT: 23

Average WT: 5.75

Total TAT: 41

Average TAT: 10.25

Gantt Chart:

	P3		P4		P2		P1	
0	4	10	15	18				

==== Code Execution Successful ===