**EXPERIMENT -6**

6. Construct a C program to implement pre-emptive priority scheduling algorithm.

#include <stdio.h>

#define MAX 100

struct Process {

int id;

int arrival\_time;

int burst\_time;

int remaining\_time;

int priority;

int completion\_time;

int waiting\_time;

int turnaround\_time;

int is\_completed;

};

int main() {

struct Process p[MAX];

int n, time = 0, completed = 0;

float total\_waiting\_time = 0, total\_turnaround\_time = 0;

printf("Enter the number of processes: ");

scanf("%d", &n);

printf("Enter arrival time, burst time, and priority (lower value = higher priority) for each process:\n");

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

p[i].id = i + 1;

printf("Process %d: ", i + 1);

scanf("%d%d%d", &p[i].arrival\_time, &p[i].burst\_time, &p[i].priority);

p[i].remaining\_time = p[i].burst\_time;

p[i].is\_completed = 0;

}

printf("\n--- Gantt Chart ---\n");

while (completed != n) {

int idx = -1;

int highest\_priority = 1e9;

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

if (p[i].arrival\_time <= time && !p[i].is\_completed && p[i].priority < highest\_priority && p[i].remaining\_time > 0) {

highest\_priority = p[i].priority;

idx = i;

}

if (idx != -1) {

printf("| P%d ", p[idx].id);

p[idx].remaining\_time--;

time++;

if (p[idx].remaining\_time == 0) {

p[idx].completion\_time = time;

p[idx].turnaround\_time = p[idx].completion\_time - p[idx].arrival\_time;

p[idx].waiting\_time = p[idx].turnaround\_time - p[idx].burst\_time;

p[idx].is\_completed = 1;

total\_waiting\_time += p[idx].waiting\_time;

total\_turnaround\_time += p[idx].turnaround\_time;

completed++;

}

} else {

printf("| idle ");

time++;

}

}

printf("|\n");

printf("\nProcess\tAT\tBT\tPri\tCT\tTAT\tWT\n");

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

printf("P%d\t%d\t%d\t%d\t%d\t%d\t%d\n", p[i].id, p[i].arrival\_time, p[i].burst\_time, p[i].priority,

p[i].completion\_time, p[i].turnaround\_time, p[i].waiting\_time);

}

printf("\nAverage Waiting Time: %.2f\n", total\_waiting\_time / n);

printf("Average Turnaround Time: %.2f\n", total\_turnaround\_time / n);

return 0;

}

**SAMPLE INPUT:**

Enter the number of processes: 3

Enter arrival time, burst time, and priority (lower value = higher priority) for each process:

Process 1: 0 5 2

Process 2: 1 3 1

Process 3: 2 1 3

**SAMPLE OUTPUT:**

**--- Gantt Chart ---**

**| P1 | P2 | P2 | P2 | P1 | P1 | P1 | P3 |**

**Process AT BT Pri CT TAT WT**

**P1 0 5 2 7 7 2**

**P2 1 3 1 4 3 0**

**P3 2 1 3 8 6 5**

**Average Waiting Time: 2.33**

**Average Turnaround Time: 5.33**