5. Write a program to compute the average waiting time and turnaround time based on Preemptive shortest remaining processing time first (SRPT) algorithm for the following set of processes, with the arrival times and the CPU-burst times given in milliseconds

Process Arrival Time Burst Time

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 |  | 0 |  | 5 |
| P2 |  | 1 |  | 3 |
| P3 |  | 2 |  | 3 |
| P4 |  | 4 |  | 1 |

Program: #include <stdio.h> struct process { int arrival\_time; int burst\_time; int remaining\_time; int waiting\_time; int turnaround\_time; int completed;

}; int main() {

int n = 4, t = 0, min\_burst\_time, min\_index; struct process processes[] = {

{0, 5, 5, 0, 0, 0},

{1, 3, 3, 0, 0, 0},

{2, 3, 3, 0, 0, 0},

{4, 1, 1, 0, 0, 0}

}; while (1) {

min\_burst\_time = 9999; min\_index = -1;

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

if (processes[i].arrival\_time <= t && processes[i].completed == 0) { if (processes[i].remaining\_time < min\_burst\_time) { min\_burst\_time = processes[i].remaining\_time; min\_index = i;

}

}

}

if (min\_index == -1) {

break;

}

processes[min\_index].remaining\_time--;

t++;

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

if (processes[i].arrival\_time <= t && processes[i].completed == 0) { if (i != min\_index) {

processes[i].waiting\_time++;

}

if (processes[i].remaining\_time == 0) { processes[i].completed = 1;

processes[i].turnaround\_time = t - processes[i].arrival\_time;

}

}

}

}

float avg\_waiting\_time = 0, avg\_turnaround\_time = 0; for (int i = 0; i < n; i++) {

avg\_waiting\_time += processes[i].waiting\_time; avg\_turnaround\_time += processes[i].turnaround\_time;

}

avg\_waiting\_time /= n; avg\_turnaround\_time /= n; printf("Average Waiting Time: %.2f ms\n", avg\_waiting\_time); printf("Average Turnaround Time: %.2f ms\n", avg\_turnaround\_time);

return 0; }

Output:

