17. Write a program to compute the average waiting time and average turnaround time based on Round Robin scheduling for the following process with the given CPU burst times and quantum time slots 4 ms, ( and the assumption that all jobs arrive at the same time.)

Process Burst Time

P1 24

P2 3

P3 3

#include<stdio.h>

int main() {

int n = 3; // Number of processes

int bt[] = {24, 3, 3}; // Burst times of the processes

int quantum = 4; // Quantum time slot

int wt[n], tat[n], rem\_bt[n], time = 0;

float avg\_wt = 0, avg\_tat = 0;

// Initialize remaining burst time to be equal to the burst time of each process

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

rem\_bt[i] = bt[i];

}

// Run the round robin scheduling algorithm

while (1) {

int done = 1;

// Traverse all processes one by one repeatedly

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

// If the remaining burst time of a process is greater than 0, then it is not yet completed

if (rem\_bt[i] > 0) {

done = 0;

// If the remaining burst time of a process is less than or equal to the quantum time slot, then it can complete in this time slot itself

if (rem\_bt[i] <= quantum) {

time += rem\_bt[i];

tat[i] = time;

rem\_bt[i] = 0;

}

// If the remaining burst time of a process is greater than the quantum time slot, then it can execute for the quantum time slot and then be moved to the end of the queue

else {

time += quantum;

rem\_bt[i] -= quantum;

}

}

}

// If all processes are done, then exit the loop

if (done == 1) {

break;

}

}

// Calculate the waiting time and turnaround time of each process

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

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

avg\_wt += wt[i];

avg\_tat += tat[i];

}

// Calculate the average waiting time and average turnaround time

avg\_wt /= n;

avg\_tat /= n;

// Print the results

printf("Process\tBurst Time\tWaiting Time\tTurnaround Time\n");

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

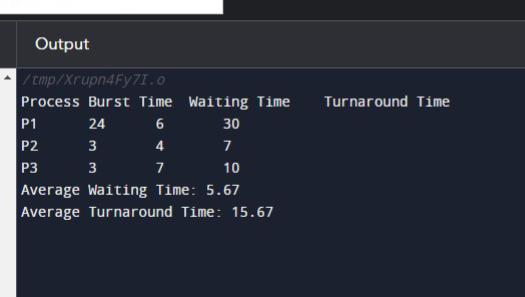
printf("P%d\t\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);

}

printf("Average Waiting Time: %.2f\n", avg\_wt);

printf("Average Turnaround Time: %.2f\n", avg\_tat);

return 0;

}