Practical 5

In an operating system three CPU-intensive processes are ready for execution, which require 10ns,20ns and 30ns and arrival at times Qns,2ns and 6ns, respectively. Write a program to calculate the total number of context switches needed if the operating system implements a shortest job first (preemptive) scheduling algorithm. Also calculate the average time for which the processes have to wait before getting the CPU.

main.c Share Run

```c
#include <stdio.h>
#include <limits.h>
#define N 3   // Number of processes
int main() {
    int arrival[N] = {0, 2, 6};
    int burst[N]   = {10, 20, 30};
    int remaining[N];
    int completion[N];
    int waiting[N] = {0};
    int time = 0, completed = 0;
    int shortest, min_time;
    int context_switches = 0;
    int prev_process = -1;
    // Initialize remaining time
    for(int i = 0; i < N; i++) {
        remaining[i] = burst[i];
    }
    while(completed != N) {
        shortest = -1;
        min_time = INT_MAX;
        // Find process with minimum remaining time
        for(int i = 0; i < N; i++) {
            if(arrival[i] <= time && remaining[i] > 0 &&
                remaining[i] < min_time) {
                min_time = remaining[i];
                shortest = i;
            }
        }
        if(shortest == -1) {
            time++;
            continue;
        }
        // Count context switch
        if(prev_process != -1 && prev_process != shortest)
            {
            context_switches++;
        }
        prev_process = shortest;
        // Execute process for 1 unit time
        remaining[shortest]--;
        time++;
        // If process completes
        if(remaining[shortest] == 0) {
            completed++;
            completion[shortest] = time;
        }
    }
    float avg_waiting = 0;
    printf("\nProcess\tArrival\tBurst\tCompletion\tWaiting
        \n");
    for(int i = 0; i < N; i++) {
        waiting[i] = completion[i] - arrival[i] - burst[i];
        avg_waiting += waiting[i];
        printf("P%d\t%d\t%d\t%d\t\t%d\n",
            i+1, arrival[i], burst[i], completion[i],
                waiting[i]);
    }
    avg_waiting /= N;
    printf("\nTotal Context Switches = %d\n",
        context_switches);
    printf("Average Waiting Time = %.2f ns\n", avg_waiting
        );
    return 0;
}
```

Output :

Output

```
Process Arrival Burst   Completion  Waiting
P1  0   10  10      0
P2  2   20  30      8
P3  6   30  60      24

Total Context Switches = 2
Average Waiting Time = 10.67 ns


=== Code Execution Successful ===
```