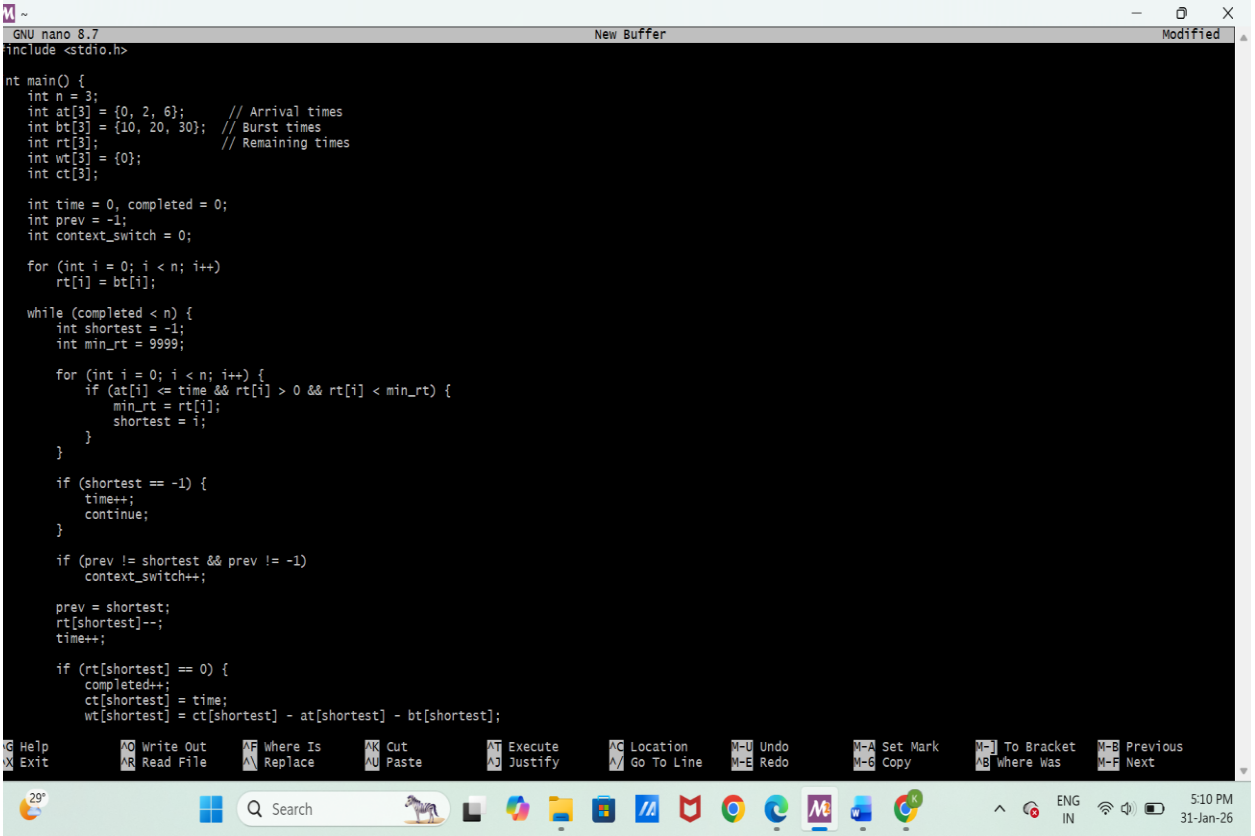
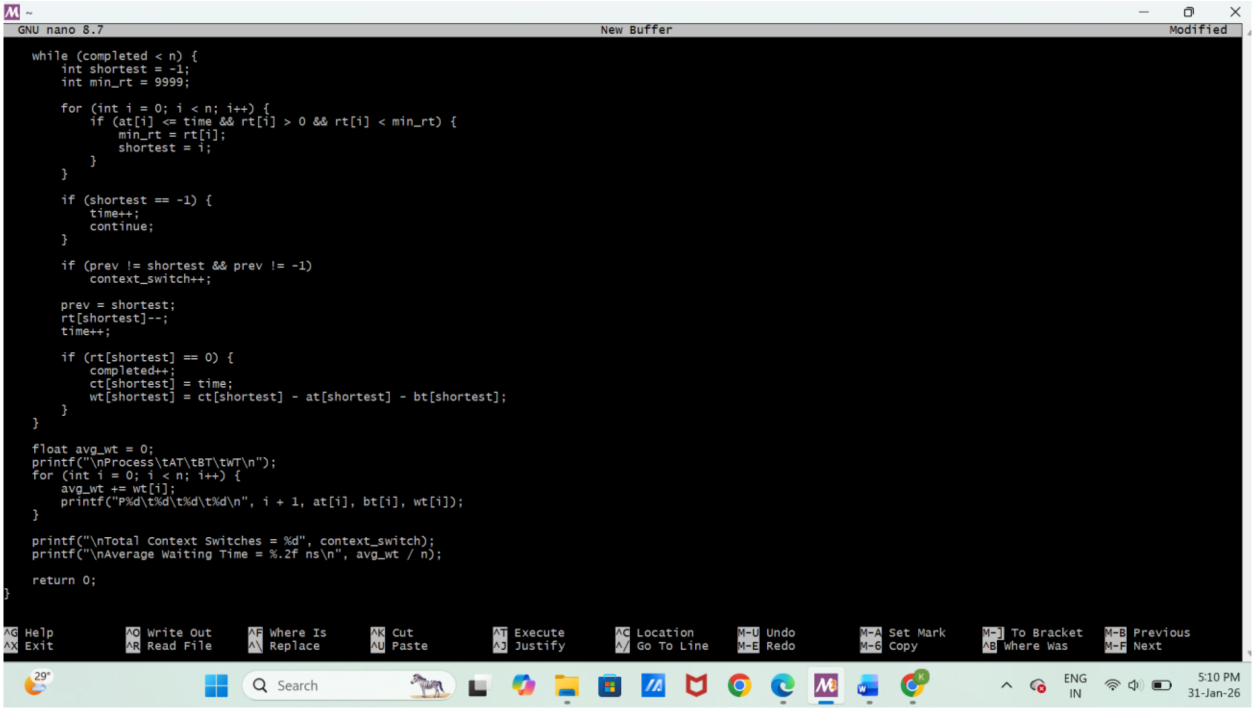
5.[Pracess management Based practical]

In an operating system three CPU-intensive processes are ready for execution, which require 10ns, 20ns and 30ns and arrival at times Ons, 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.

CODE:

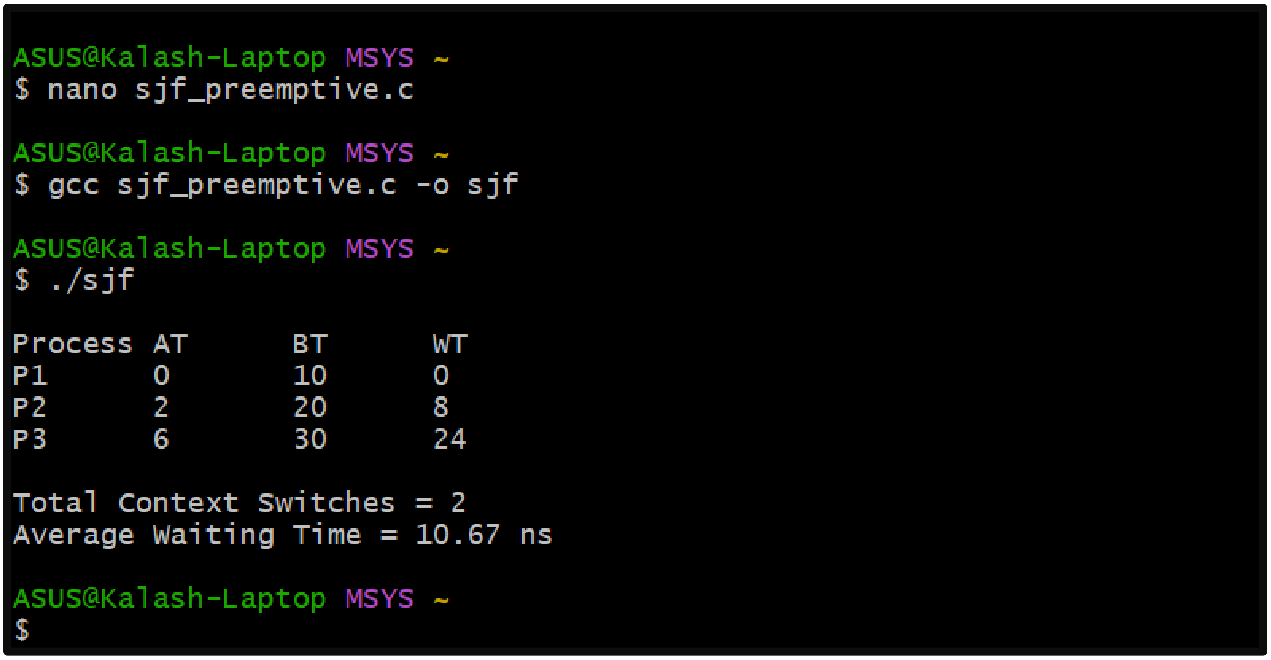


```
#include <stdio.h>

nt main() {
   int n = 3;
   int at[3] = {0, 2, 6};      // Arrival times
   int bt[3] = {10, 20, 30};  // Burst times
   int rt[3];                 // Remaining times
   int wt[3] = {0};
   int ct[3];

   int time = 0, completed = 0;
   int prev = -1;
   int context_switch = 0;

   for (int i = 0; i < n; i++)
       rt[i] = bt[i];

   while (completed < n) {
       int shortest = -1;
       int min_rt = 9999;

       for (int i = 0; i < n; i++) {
           if (at[i] <= time && rt[i] > 0 && rt[i] < min_rt) {
               min_rt = rt[i];
               shortest = i;
           }
       }

       if (shortest == -1) {
           time++;
           continue;
       }

       if (prev != shortest && prev != -1)
           context_switch++;

       prev = shortest;
       rt[shortest]--;
       time++;

       if (rt[shortest] == 0) {
           completed++;
           ct[shortest] = time;
           wt[shortest] = ct[shortest] - at[shortest] - bt[shortest];
```



```c
    while (completed < n) {
        int shortest = -1;
        int min_rt = 9999;

        for (int i = 0; i < n; i++) {
            if (at[i] <= time && rt[i] > 0 && rt[i] < min_rt) {
                min_rt = rt[i];
                shortest = i;
            }
        }

        if (shortest == -1) {
            time++;
            continue;
        }

        if (prev != shortest && prev != -1)
            context_switch++;

        prev = shortest;
        rt[shortest]--;
        time++;

        if (rt[shortest] == 0) {
            completed++;
            ct[shortest] = time;
            wt[shortest] = ct[shortest] - at[shortest] - bt[shortest];
        }
    }

    float avg_wt = 0;
    printf("\nProcess\tAT\tBT\tWT\n");
    for (int i = 0; i < n; i++) {
        avg_wt += wt[i];
        printf("P%d\t%d\t%d\t%d\n", i + 1, at[i], bt[i], wt[i]);
    }

    printf("\nTotal Context Switches = %d", context_switch);
    printf("\nAverage Waiting Time = %.2f ns\n", avg_wt / n);

    return 0;
}
```

Output:



```
ASUS@Kalash-Laptop MSYS ~
$ nano sjf_preemptive.c

ASUS@Kalash-Laptop MSYS ~
$ gcc sjf_preemptive.c -o sjf

ASUS@Kalash-Laptop MSYS ~
$ ./sjf

Process AT      BT      WT
P1      0       10      0
P2      2       20      8
P3      6       30      24

Total Context Switches = 2
Average Waiting Time = 10.67 ns

ASUS@Kalash-Laptop MSYS ~
$
```