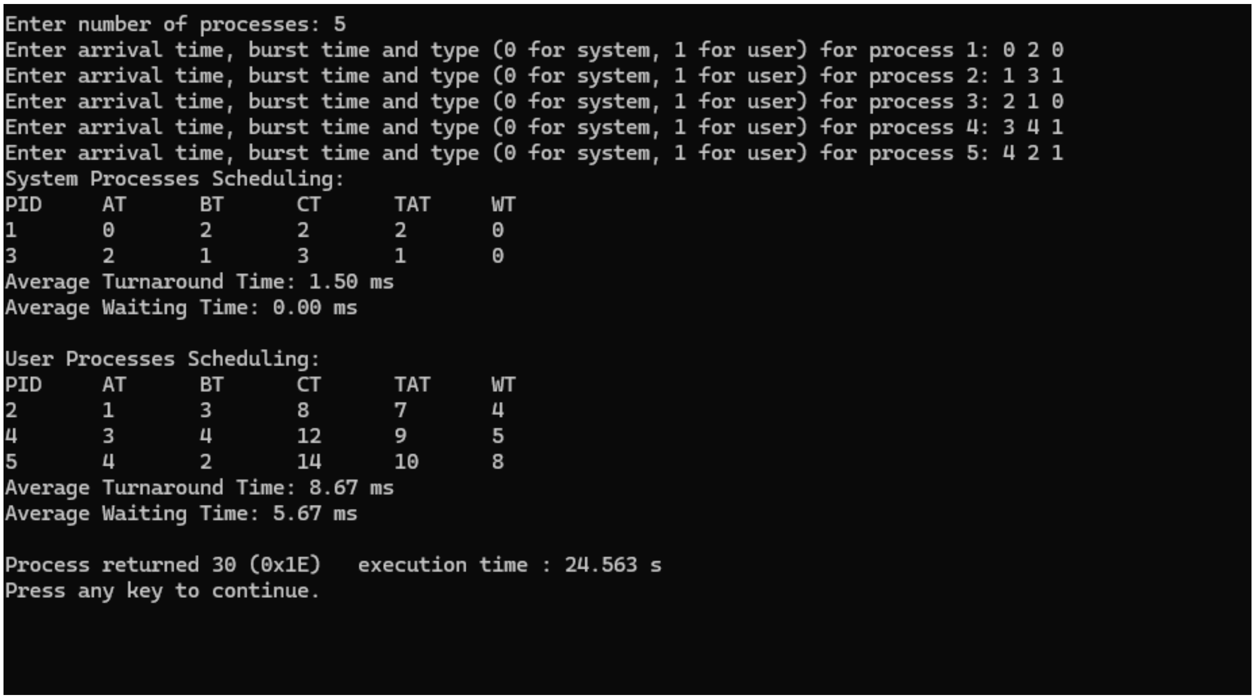
LAB 3 :
Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories – system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for the processes in each queue.

```c
#include <stdio.h>

void sort(int proc_id[], int at[], int bt[], int n) {
    int min, temp;
    for(int i=0; i<n-1; i++) {
        for(int j=i+1; j<n; j++) {
            if(at[j] < at[i]) {

                temp = at[i];
                at[i] = at[j];
                at[j] = temp;

                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;

                temp = proc_id[i];
                proc_id[i] = proc_id[j];
                proc_id[j] = temp;
            }
        }
    }
}

void simulateFCFS(int proc_id[], int at[], int bt[], int n, int start_time) {
    int c = start_time, ct[n], tat[n], wt[n];
    double ttat = 0.0, twt = 0.0;


    for(int i=0; i<n; i++) {
        if(c >= at[i])
            c += bt[i];
        else
            c = at[i] + bt[i];
        ct[i] = c;
    }
    for(int i=0; i<n; i++)
        tat[i] = ct[i] - at[i];


    for(int i=0; i<n; i++)
        wt[i] = tat[i] - bt[i];

    printf("PID\tAT\tBT\tCT\tTAT\tWT\n");
    for(int i=0; i<n; i++) {
        printf("%d\t%d\t%d\t%d\t%d\t%d\n", proc_id[i], at[i], bt[i], ct[i], tat[i], wt[i]);
        ttat += tat[i];
        twt += wt[i];
    }

    printf("Average Turnaround Time: %.2lf ms\n", ttat/n);
    printf("Average Waiting Time: %.2lf ms\n", twt/n);
}

void main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    int proc_id[n], at[n], bt[n], type[n];
    int sys_proc_id[n], sys_at[n], sys_bt[n], user_proc_id[n], user_at[n], user_bt[n];
    int sys_count = 0, user_count = 0;

    for(int i=0; i<n; i++) {
        proc_id[i] = i + 1;
        printf("Enter arrival time, burst time and type (0 for system, 1 for user) for process %d: ",
i+1);
        scanf("%d %d %d", &at[i], &bt[i], &type[i]);

        if(type[i] == 0) {
            sys_proc_id[sys_count] = proc_id[i];
            sys_at[sys_count] = at[i];
            sys_bt[sys_count] = bt[i];
            sys_count++;
        } else {
            user_proc_id[user_count] = proc_id[i];
            user_at[user_count] = at[i];
            user_bt[user_count] = bt[i];
            user_count++;
        }
    }


    sort(sys_proc_id, sys_at, sys_bt, sys_count);
    sort(user_proc_id, user_at, user_bt, user_count);


    printf("System Processes Scheduling:\n");
    simulateFCFS(sys_proc_id, sys_at, sys_bt, sys_count, 0);


    int system_end_time = 0;
    if (sys_count > 0) {
        system_end_time = sys_at[sys_count - 1] + sys_bt[sys_count - 1];
        for (int i = 0; i < sys_count - 1; i++) {
            if (sys_at[i + 1] > system_end_time) {
                system_end_time = sys_at[i + 1];
            }
            system_end_time += sys_bt[i];
        }
    }

    printf("\nUser Processes Scheduling:\n");
    simulateFCFS(user_proc_id, user_at, user_bt, user_count, system_end_time);
}
```

OUTPUT:



```
Enter number of processes: 5
Enter arrival time, burst time and type (0 for system, 1 for user) for process 1: 0 2 0
Enter arrival time, burst time and type (0 for system, 1 for user) for process 2: 1 3 1
Enter arrival time, burst time and type (0 for system, 1 for user) for process 3: 2 1 0
Enter arrival time, burst time and type (0 for system, 1 for user) for process 4: 3 4 1
Enter arrival time, burst time and type (0 for system, 1 for user) for process 5: 4 2 1
System Processes Scheduling:
PID     AT      BT      CT      TAT     WT
1       0       2       2       2       0
3       2       1       3       1       0
Average Turnaround Time: 1.50 ms
Average Waiting Time: 0.00 ms

User Processes Scheduling:
PID     AT      BT      CT      TAT     WT
2       1       3       8       7       4
4       3       4       12      9       5
5       4       2       14      10      8
Average Turnaround Time: 8.67 ms
Average Waiting Time: 5.67 ms

Process returned 30 (0x1E)   execution time : 24.563 s
Press any key to continue.
```