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.

```
#include <stdio.h>

struct Process {
    int pid, at, bt, ct, tat, wt, queueType;
};

void sortByArrival(struct Process p[], int n) {
    struct Process temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (p[j].at > p[j + 1].at) {
                temp = p[j];
                p[j] = p[j + 1];
                p[j + 1] = temp;
            }
        }
    }
}

void calculateTimes(struct Process p[], int n, int *globalTime) {
    for (int i = 0; i < n; i++) {
        if (*globalTime < p[i].at)
            *globalTime = p[i].at;

        p[i].ct = *globalTime + p[i].bt;
        p[i].tat = p[i].ct - p[i].at;
        p[i].wt = p[i].tat - p[i].bt;
        *globalTime = p[i].ct;
    }
}

void displayProcesses(struct Process p[], int n) {
    printf("\nPID\tAT\tBT\tQueue\tCT\tTAT\tWT\n");
    printf("-------------------------------------------------\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t%s\t%d\t%d\t%d\n",
```

```
            p[i].pid, p[i].at, p[i].bt,
            (p[i].queueType == 0) ? "System" : "User",
            p[i].ct, p[i].tat, p[i].wt);
    }
}

void calculateAvgTimes(struct Process p[], int n) {
    float totalTAT = 0, totalWT = 0;
    for (int i = 0; i < n; i++) {
        totalTAT += p[i].tat;
        totalWT += p[i].wt;
    }
    printf("\nAverage Turnaround Time: %.2f", totalTAT / n);
    printf("\nAverage Waiting Time: %.2f\n", totalWT / n);
}

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

    struct Process p[n], systemQueue[n], userQueue[n];
    int sysCount = 0, userCount = 0;

    for (int i = 0; i < n; i++) {
        printf("\nEnter details for process %d\n", i + 1);
        p[i].pid = i + 1;
        printf("Arrival Time: ");
        scanf("%d", &p[i].at);
        printf("Burst Time: ");
        scanf("%d", &p[i].bt);
        printf("Queue Type (0 for System, 1 for User): ");
        scanf("%d", &p[i].queueType);

        if (p[i].queueType == 0)
            systemQueue[sysCount++] = p[i];
        else
            userQueue[userCount++] = p[i];
    }


    sortByArrival(systemQueue, sysCount);
    sortByArrival(userQueue, userCount);
```

```
    int globalTime = 0;


    calculateTimes(systemQueue, sysCount, &globalTime);


    calculateTimes(userQueue, userCount, &globalTime);


    for (int i = 0; i < sysCount; i++)
        p[i] = systemQueue[i];
    for (int i = 0; i < userCount; i++)
        p[sysCount + i] = userQueue[i];

    displayProcesses(p, n);
    calculateAvgTimes(p, n);

    return 0;
}
```

/***Output:**

```
Enter the number of processes: 5

Enter details for process 1
Arrival Time: 0
Burst Time: 3
Queue Type (0 for System, 1 for User): 0

Enter details for process 2
Arrival Time: 2
Burst Time: 2
Queue Type (0 for System, 1 for User): 1

Enter details for process 3
Arrival Time: 3
Burst Time: 5
Queue Type (0 for System, 1 for User): 0

Enter details for process 4
Arrival Time: 4
Burst Time: 4
Queue Type (0 for System, 1 for User): 1
```

Enter details for process 5
Arrival Time: 5
Burst Time: 1
Queue Type (0 for System, 1 for User): 0

PID	AT	BT	Queue	CT	TAT	WT
1	0	3	System	3	3	0
3	3	5	System	8	5	0
5	5	1	System	9	4	3
2	2	2	User	11	9	7
4	4	4	User	15	11	7

Average Turnaround Time: 6.40
Average Waiting Time: 3.40
*/