LAB PROGRAM - 04

Write a C program to simulate a 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>
#include <stdlib.h>
#include <string.h>

typedef struct {
    int process_id;
    char type[10];
    int arrival_time;
    int burst_time;
} Process;

void sortByArrivalTime(Process processes[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (processes[j].arrival_time > processes[j + 1].arrival_time)
{
                Process temp = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = temp;
            }
        }
    }
}
void fcfsScheduling(Process queue[], int n, const char *queue_name) {
    printf("\nScheduling processes in %s queue:\n", queue_name);
    int time = 0;

    for (int i = 0; i < n; i++) {
        if (time < queue[i].arrival_time)
            time = queue[i].arrival_time;

        printf("Process %d (Arrival: %d, Burst: %d) starts at %d and ends
at %d.\n",
```

```
               queue[i].process_id, queue[i].arrival_time,
queue[i].burst_time, time,
               time + queue[i].burst_time);
        time += queue[i].burst_time;
    }
}

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

    Process processes[n];
    Process systemQueue[n], userQueue[n];
    int systemCount = 0, userCount = 0;

    for (int i = 0; i < n; i++) {
        printf("Enter process ID, type (system/user), arrival time, and
burst time for process %d: ", i + 1);
        scanf("%d %s %d %d", &processes[i].process_id, processes[i].type,
&processes[i].arrival_time, &processes[i].burst_time);

        if (strcmp(processes[i].type, "system") == 0) {
            systemQueue[systemCount++] = processes[i];
        } else if (strcmp(processes[i].type, "user") == 0) {
            userQueue[userCount++] = processes[i];
        }
    }

    sortByArrivalTime(systemQueue, systemCount);
    sortByArrivalTime(userQueue, userCount);

    fcfsScheduling(systemQueue, systemCount, "System");
    fcfsScheduling(userQueue, userCount, "User");

    return 0;
}
```

Output

```
PS C:\Users\Admin\Downloads> cd "c:\Users\Admin\Downloads\" ; if ($?) { gcc Untitled-2.c -o Untitled
Enter the number of processes: 2
Enter process ID, type (system/user), arrival time, and burst time for process 1: 1 user 2 4
Enter process ID, type (system/user), arrival time, and burst time for process 2: 2 system 4 2

Scheduling processes in System queue:
Process 2 (Arrival: 4, Burst: 2) starts at 4 and ends at 6.

Scheduling processes in User queue:
Process 1 (Arrival: 2, Burst: 4) starts at 2 and ends at 6.
PS C:\Users\Admin\Downloads>
```