Program-2

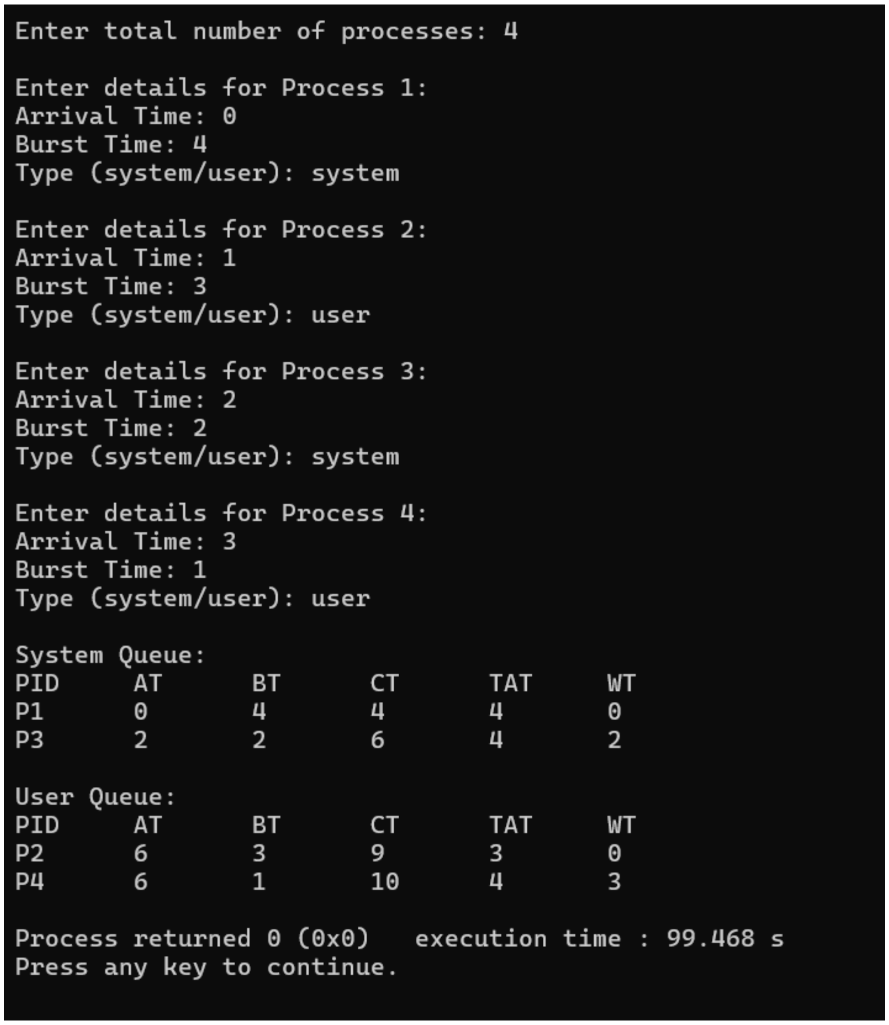
Write a C program to simulate multi-level queue scheduling algorithm considering the following scenario. All the processes in the system are divided into two categories –system processes and user processes. System processes are to be given higher priority than user processes. Use FCFS scheduling for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

Code:

```
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100

typedef struct {
    int pid;
    int arrival_time;
    int burst_time;
    int completion_time;
    int turnaround_time;
    int waiting_time;
    char type[10]; // "system" or "user"
} Process;

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

void calculate_times(Process p[], int n) {
    int current_time = 0;
    for (int i = 0; i < n; i++) {
        if (current_time < p[i].arrival_time)
```

```
            current_time = p[i].arrival_time;

        p[i].completion_time = current_time + p[i].burst_time;
        p[i].turnaround_time = p[i].completion_time - p[i].arrival_time;
        p[i].waiting_time = p[i].turnaround_time - p[i].burst_time;
        current_time = p[i].completion_time;
    }
}

void print_processes(Process p[], int n, const char *queue_name) {
    printf("\n%s Queue:\n", queue_name);
    printf("PID\tAT\tBT\tCT\tTAT\tWT\n");
    for (int i = 0; i < n; i++) {
        printf("P%d\t%d\t%d\t%d\t%d\t%d\n",
               p[i].pid,
               p[i].arrival_time,
               p[i].burst_time,
               p[i].completion_time,
               p[i].turnaround_time,
               p[i].waiting_time);
    }
}

int main() {
    int n;
    Process system_queue[MAX], user_queue[MAX];
    int sys_count = 0, user_count = 0;

    printf("Enter total number of processes: ");
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        Process temp;
        temp.pid = i + 1;

        printf("\nEnter details for Process %d:\n", temp.pid);
        printf("Arrival Time: ");
        scanf("%d", &temp.arrival_time);
        printf("Burst Time: ");
        scanf("%d", &temp.burst_time);
        printf("Type (system/user): ");
        scanf("%s", temp.type);

        if (strcmp(temp.type, "system") == 0) {
            system_queue[sys_count++] = temp;
        } else if (strcmp(temp.type, "user") == 0) {
            user_queue[user_count++] = temp;
        } else {
            printf("Invalid type! Skipping process.\n");
        }
    }

    // Sort both queues by arrival time
    sort_by_arrival(system_queue, sys_count);
    sort_by_arrival(user_queue, user_count);

    // Calculate times for each queue
    calculate_times(system_queue, sys_count);

    // Calculate starting time for user queue
    int last_completion_time = 0;
    if (sys_count > 0)
        last_completion_time = system_queue[sys_count - 1].completion_time;

    // Adjust user arrival time to account for idle time if any
    for (int i = 0; i < user_count; i++) {
        if (user_queue[i].arrival_time < last_completion_time)
            user_queue[i].arrival_time = last_completion_time;
    }

    // Sort again in case arrival times changed
    sort_by_arrival(user_queue, user_count);

    // Calculate times for user queue
    calculate_times(user_queue, user_count);

    // Print results
    print_processes(system_queue, sys_count, "System");
    print_processes(user_queue, user_count, "User");

    return 0;
}
```

Output:



```
Enter total number of processes: 4

Enter details for Process 1:
Arrival Time: 0
Burst Time: 4
Type (system/user): system

Enter details for Process 2:
Arrival Time: 1
Burst Time: 3
Type (system/user): user

Enter details for Process 3:
Arrival Time: 2
Burst Time: 2
Type (system/user): system

Enter details for Process 4:
Arrival Time: 3
Burst Time: 1
Type (system/user): user

System Queue:
PID     AT      BT      CT      TAT     WT
P1      0       4       4       4       0
P3      2       2       6       4       2

User Queue:
PID     AT      BT      CT      TAT     WT
P2      6       3       9       3       0
P4      6       1       10      4       3

Process returned 0 (0x0)   execution time : 99.468 s
Press any key to continue.
```