# e. Multilevel queue

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_PROC 10

typedef struct {
    int procID;
    int arrivalTime;
    int burstTime;
    int remainingTime;
    int isCompleted;
} Task;

void FCFS_Scheduling(Task queue[], int total) {
    int currentTime = 0;
    for (int i = 0; i < total; i++) {
        if (queue[i].arrivalTime > currentTime)
            currentTime = queue[i].arrivalTime;
        printf("Task %d starts at time %d and finishes at time %d\n", queue[i].procID, currentTime, curre
        currentTime += queue[i].burstTime;
        queue[i].isCompleted = 1;
    }
}

void RoundRobin_Scheduling(Task queue[], int total, int quantumTime) {
    int currentTime = 0;
    int completedCount = 0;
    int idx = 0;
    while (completedCount < total) {
        if (queue[idx].isCompleted == 0 && queue[idx].remainingTime > 0) {
            int executionTime = queue[idx].remainingTime > quantumTime ? quantumTime : queue[idx].re
            printf("Task %d runs from time %d to %d (Quantum: %d)\n", queue[idx].procID, currentTime,
            currentTime += executionTime;
            queue[idx].remainingTime -= executionTime;

            if (queue[idx].remainingTime == 0) {
                queue[idx].isCompleted = 1;
                completedCount++;
            }
        }
        idx = (idx + 1) % total;
    }
```

```c
}

int main() {
    int totalTasks, i, timeQuantum;
    printf("Enter the number of tasks: ");
    scanf("%d", &totalTasks);
    Task highPriorityQueue[MAX_PROC], lowPriorityQueue[MAX_PROC];
    int highCount = 0, lowCount = 0;

    for (i = 0; i < totalTasks; i++) {
        printf("\nEnter details for Task %d:\n", i + 1);
        printf("Arrival Time: ");
        scanf("%d", &highPriorityQueue[i].arrivalTime);
        printf("Burst Time: ");
        scanf("%d", &highPriorityQueue[i].burstTime);
        printf("Priority (1 for high, 2 for low): ");
        int priority;
        scanf("%d", &priority);

        highPriorityQueue[i].procID = i + 1;
        highPriorityQueue[i].remainingTime = highPriorityQueue[i].burstTime;
        highPriorityQueue[i].isCompleted = 0;

        if (priority == 1) {
            highPriorityQueue[highCount++] = highPriorityQueue[i];
        } else if (priority == 2) {
            lowPriorityQueue[lowCount++] = highPriorityQueue[i];
        }
    }

    printf("\nEnter the time quantum for Round Robin scheduling: ");
    scanf("%d", &timeQuantum);

    printf("\nHigh-priority queue (FCFS scheduling):\n");
    FCFS_Scheduling(highPriorityQueue, highCount);

    printf("\nLow-priority queue (Round Robin scheduling):\n");
    RoundRobin_Scheduling(lowPriorityQueue, lowCount, timeQuantum);

    return 0;
}
```

```
Enter the number of tasks: 3
Enter details for Task 1:
Arrival Time: 0
Burst Time: 5
Priority (1 for high, 2 for low): 1

Enter details for Task 2:
Arrival Time: 1
Burst Time: 3
Priority (1 for high, 2 for low): 2

Enter details for Task 3:
Arrival Time: 2
Burst Time: 2
Priority (1 for high, 2 for low): 1

Enter the time quantum for Round Robin scheduling: 2

High-priority queue (FCFS scheduling):
Task 1 starts at time 0 and finishes at time 5
Task 3 starts at time 5 and finishes at time 7

Low-priority queue (Round Robin scheduling):
Task 2 runs from time 7 to 9 (Quantum: 2)
Task 2 runs from time 9 to 10 (Quantum: 1)
```