f. Rate Monotonic

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MAX_COUNT 5

typedef struct {
    int taskID;
    int cycleTime;
    int execTime;
} ScheduledTask;

int compareTasks(const void *a, const void *b) {
    ScheduledTask *taskA = (ScheduledTask *)a;
    ScheduledTask *taskB = (ScheduledTask *)b;
    return taskA->cycleTime - taskB->cycleTime;
}

void executeRateMonotonic(ScheduledTask tasks[], int taskCount, int maxTime) {
    int currentTime = 0;

    while (currentTime < maxTime) {
        for (int i = 0; i < taskCount; i++) {
            if (currentTime % tasks[i].cycleTime == 0) {
                printf("At time %d ms: Task %d (Period: %d ms) is executing for %d ms\n", currentTime, tas
                usleep(tasks[i].execTime * 1000);
            }
        }
        currentTime++;
    }
}

int main() {

    ScheduledTask tasks[MAX_COUNT] = {
        {1, 5, 1},
        {2, 10, 2},
        {3, 15, 3},
        {4, 20, 4},
        {5, 25, 5}
    };
```

```
    int taskCount = 5;
    int maxTime = 50;

    qsort(tasks, taskCount, sizeof(ScheduledTask), compareTasks);

    executeRateMonotonic(tasks, taskCount, maxTime);

    return 0;
}
```

```
At time 0 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 5 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 10 ms: Task 2 (Period: 10 ms) is executing for 2 ms
At time 10 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 15 ms: Task 3 (Period: 15 ms) is executing for 3 ms
At time 15 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 20 ms: Task 4 (Period: 20 ms) is executing for 4 ms
At time 20 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 25 ms: Task 5 (Period: 25 ms) is executing for 5 ms
At time 25 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 30 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 35 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 40 ms: Task 1 (Period: 5 ms) is executing for 1 ms
At time 45 ms: Task 1 (Period: 5 ms) is executing for 1 ms
```