LAB PROGRAM - 04

Write a C program to simulate Real-Time CPU Scheduling algorithms:
a) Rate- Monotonic
b) Earliest-deadline First

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

#define MAX_TASKS 10

typedef struct {
    int id;
    int execution_time;
    int period;
    int deadline;
    int remaining_time;
    int next_arrival;
} Task;

void sort_by_period(Task tasks[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (tasks[j].period > tasks[j + 1].period) {
                Task temp = tasks[j];
                tasks[j] = tasks[j + 1];
                tasks[j + 1] = temp;
            }
        }
    }
}

void sort_by_deadline(Task tasks[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (tasks[j].deadline > tasks[j + 1].deadline) {
                Task temp = tasks[j];
                tasks[j] = tasks[j + 1];
                tasks[j + 1] = temp;
            }
```

```
        }
    }
}
void simulate_RMS(Task tasks[], int n, int simulation_time) {
    sort_by_period(tasks, n);

    int current_time = 0;
    printf("\nRate-Monotonic Scheduling Execution:\n");

    while (current_time < simulation_time) {
        int selected_task = -1;

        for (int i = 0; i < n; i++) {
            if (tasks[i].remaining_time > 0 && current_time >=
tasks[i].next_arrival) {
                selected_task = i;
                break;
            }
        }

        if (selected_task != -1) {
            printf("Time %d - Task %d executing\n", current_time,
tasks[selected_task].id);
            tasks[selected_task].remaining_time--;

            if (tasks[selected_task].remaining_time == 0) {
                tasks[selected_task].remaining_time =
tasks[selected_task].execution_time;
                tasks[selected_task].next_arrival +=
tasks[selected_task].period;
            }
        } else {
            printf("Time %d - Idle\n", current_time);
        }
        current_time++;
    }
}

void simulate_EDF(Task tasks[], int n, int simulation_time) {
    int current_time = 0;
```

```
    printf("\nEarliest Deadline First Scheduling Execution:\n");

    while (current_time < simulation_time) {

        for (int i = 0; i < n; i++) {
            if (current_time >= tasks[i].next_arrival) {
                tasks[i].deadline = current_time + tasks[i].period;
                tasks[i].remaining_time = tasks[i].execution_time;
                tasks[i].next_arrival += tasks[i].period;
            }
        }

        sort_by_deadline(tasks, n);

        int selected_task = -1;
        for (int i = 0; i < n; i++) {
            if (tasks[i].remaining_time > 0) {
                selected_task = i;
                break;
            }
        }

        if (selected_task != -1) {
            printf("Time %d - Task %d executing\n", current_time,
tasks[selected_task].id);
            tasks[selected_task].remaining_time--;

            if (tasks[selected_task].remaining_time == 0) {
                tasks[selected_task].deadline = 99999;
            }
        } else {
            printf("Time %d - Idle\n", current_time);
        }

        current_time++;
    }
}

int main() {
    int n, simulation_time, choice;
```

```
    printf("Enter number of tasks: ");
    scanf("%d", &n);

    Task tasks[MAX_TASKS];

    for (int i = 0; i < n; i++) {
        printf("Enter execution time and period for Task %d: ", i + 1);
        scanf("%d %d", &tasks[i].execution_time, &tasks[i].period);
        tasks[i].id = i + 1;
        tasks[i].remaining_time = tasks[i].execution_time;
        tasks[i].next_arrival = 0;
        tasks[i].deadline = tasks[i].period;
    }

    printf("Enter simulation time: ");
    scanf("%d", &simulation_time);

    printf("Choose Scheduling Algorithm:\n1. Rate-Monotonic Scheduling
(RMS)\n2. Earliest-Deadline First (EDF)\nEnter choice: ");
    scanf("%d", &choice);

    if (choice == 1) {
        simulate_RMS(tasks, n, simulation_time);
    } else if (choice == 2) {
        simulate_EDF(tasks, n, simulation_time);
    } else {
        printf("Invalid choice!\n");
    }

    return 0;
}
```

Output:

```
PS C:\Users\Admin\Downloads> cd "c:\Users\Admin\Downloac
Enter number of tasks: 2
Enter execution time and period for Task 1: 1 4
Enter execution time and period for Task 2: 2 6
Enter simulation time: 12
Choose Scheduling Algorithm:
1. Rate-Monotonic Scheduling (RMS)
2. Earliest-Deadline First (EDF)
Enter choice: 1

Rate-Monotonic Scheduling Execution:
Time 0 - Task 1 executing
Time 1 - Task 2 executing
Time 2 - Task 2 executing
Time 3 - Idle
Time 4 - Task 1 executing
Time 5 - Idle
Time 6 - Task 2 executing
Time 7 - Task 2 executing
Time 8 - Task 1 executing
Time 9 - Idle
Time 10 - Idle
Time 11 - Idle
```

```
PS C:\Users\Admin\Downloads> cd "c:\Users\Admin\Download
Enter number of tasks: 2
Enter execution time and period for Task 1: 1 4
Enter execution time and period for Task 2: 2 6
Enter simulation time: 12
Choose Scheduling Algorithm:
1. Rate-Monotonic Scheduling (RMS)
2. Earliest-Deadline First (EDF)
Enter choice: 2

Earliest Deadline First Scheduling Execution:
Time 0 - Task 1 executing
Time 1 - Task 2 executing
Time 2 - Task 2 executing
Time 3 - Idle
Time 4 - Task 1 executing
Time 5 - Idle
Time 6 - Task 2 executing
Time 7 - Task 2 executing
Time 8 - Task 1 executing
Time 9 - Idle
Time 10 - Idle
Time 11 - Idle
PS C:\Users\Admin\Downloads>
```