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

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

#define MAX_TASKS 10

typedef struct {
    int id;
    int period;
    int deadline;
    int computation_time;
    int remaining_time;
} Task;

void rate_monotonic(Task tasks[], int num_tasks);
void earliest_deadline_first(Task tasks[], int num_tasks);
void proportional_scheduling(Task tasks[], int num_tasks);

int main() {
    int num_tasks;
    Task tasks[MAX_TASKS];

    printf("Enter the number of tasks: ");
    scanf("%d", &num_tasks);

    printf("Enter the details of each task (id, period, deadline, computation time):\n");
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d: ", i + 1);
        scanf("%d %d %d %d", &tasks[i].id, &tasks[i].period, &tasks[i].deadline, &tasks[i].computation_time);
        tasks[i].remaining_time = tasks[i].computation_time;
    }
```

```
    }

    rate_monotonic(tasks, num_tasks);
    earliest_deadline_first(tasks, num_tasks);
    proportional_scheduling(tasks, num_tasks);

    return 0;
}

void rate_monotonic(Task tasks[], int num_tasks) {
    printf("\nRate-Monotonic Scheduling:\n");
    for (int i = 0; i < num_tasks - 1; i++) {
        for (int j = 0; j < num_tasks - i - 1; j++) {
            if (tasks[j].period > tasks[j + 1].period) {
                Task temp = tasks[j];
                tasks[j] = tasks[j + 1];
                tasks[j + 1] = temp;
            }
        }
    }
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d scheduled\n", tasks[i].id);
    }
}

void earliest_deadline_first(Task tasks[], int num_tasks) {
    printf("\nEarliest-Deadline First Scheduling:\n");
    for (int i = 0; i < num_tasks - 1; i++) {
        for (int j = 0; j < num_tasks - i - 1; j++) {
            if (tasks[j].deadline > tasks[j + 1].deadline) {
```

```
    }
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d scheduled\n", tasks[i].id);
    }
}

void earliest_deadline_first(Task tasks[], int num_tasks) {
    printf("\nEarliest-Deadline First Scheduling:\n");
    for (int i = 0; i < num_tasks - 1; i++) {
        for (int j = 0; j < num_tasks - i - 1; j++) {
            if (tasks[j].deadline > tasks[j + 1].deadline) {
                Task temp = tasks[j];
                tasks[j] = tasks[j + 1];
                tasks[j + 1] = temp;
            }
        }
    }
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d scheduled\n", tasks[i].id);
    }
}

void proportional_scheduling(Task tasks[], int num_tasks) {
    printf("\nProportional Scheduling:\n");
    float total_period_inverse = 0;
    for (int i = 0; i < num_tasks; i++) {
        total_period_inverse += 1.0 / tasks[i].period;
    }
    for (int i = 0; i < num_tasks; i++) {
        float proportion = (1.0 / tasks[i].period) / total_period_inverse;
        printf("Task %d gets %.2f%% of CPU time\n", tasks[i].id, proportion * 100);
    }
}

```

Output:

```
Enter the number of tasks: 3
Enter the details of each task (id, period, deadline, computation time):
Task 1: 1
5
5
2
Task 2: 2
10
10
3
Task 3: 3
15
15
4

Rate-Monotonic Scheduling:
Task 1 scheduled
Task 2 scheduled
Task 3 scheduled

Earliest-Deadline First Scheduling:
Task 1 scheduled
Task 2 scheduled
Task 3 scheduled

Proportional Scheduling:
Task 1 gets 54.55% of CPU time
Task 2 gets 27.27% of CPU time
Task 3 gets 18.18% of CPU time
```