1.write a c program to simulate real time cpu scheduling algorithm for
a. rate monotonic

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

#define MAX_TASKS 10

typedef struct {
    int period;
    int execution_time;
    int priority;
} Task;

Task tasks[MAX_TASKS];

int gcd(int a, int b) {
    if (b == 0)
        return a;
    else
        return gcd(b, a % b);
}

int lcm(int a, int b) {
    return a * b / gcd(a, b);
}

int hyperperiod(Task tasks[], int num_tasks) {
    int lcm_value = tasks[0].period;
    for (int i = 1; i < num_tasks; i++) {
        lcm_value = lcm(lcm_value, tasks[i].period);
    }
    return lcm_value;
}

void schedule(Task tasks[], int num_tasks) {
    int hyper = hyperperiod(tasks, num_tasks);
    int i, j;

    printf("Hyperperiod: %d\n", hyper);
    printf("Scheduling:\n");

    for (i = 0; i < hyper; i++) {
        int min_priority = 999;
```

```
        int task_to_execute = -1;

        for (j = 0; j < num_tasks; j++) {
            if (i % tasks[j].period == 0 && tasks[j].priority < min_priority) {
                min_priority = tasks[j].priority;
                task_to_execute = j;
            }
        }

        if (task_to_execute != -1) {
            printf("At time %d: Task %d executed\n", i, task_to_execute);
            tasks[task_to_execute].execution_time--;

            if (tasks[task_to_execute].execution_time == 0) {
                tasks[task_to_execute].execution_time = tasks[task_to_execute].period;
            }
        }
    }
}

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

    printf("Enter period and execution time for each task:\n");
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d:\n", i);
        printf("Period: ");
        scanf("%d", & tasks[i].period);
        printf("Execution time: ");
        scanf("%d", & tasks[i].execution_time);
        tasks[i].priority = i + 1;
    }

    schedule(tasks, num_tasks);

    return 0;
}
```

"C:\Users\yp671\OneDrive\Desktop\opetrating system\ratemonotonic.exe"

```
Enter number of tasks: 3
Enter period and execution time for each task:
Task 0:
Period: 20
Execution time: 3
Task 1:
Period: 15
Execution time: 2
Task 2:
Period: 5
Execution time: 2
Hyperperiod: 60
Scheduling:
At time 0: Task 0 executed
At time 5: Task 2 executed
At time 10: Task 2 executed
At time 15: Task 1 executed
At time 20: Task 0 executed
At time 25: Task 2 executed
At time 30: Task 1 executed
At time 35: Task 2 executed
At time 40: Task 0 executed
At time 45: Task 1 executed
At time 50: Task 2 executed
At time 55: Task 2 executed

Process returned 0 (0x0)   execution time : 38.933 s
Press any key to continue.
```

B.earliest deadline first

```
#include <stdio.h>
#define MAX_TASKS 10
typedef struct {
    int deadline;
    int execution_time;
    int task_id;
} Task;
void edf_schedule(Task tasks[], int num_tasks) {
    int current_time = 0;
    int remaining_time[MAX_TASKS];
    for (int i = 0; i < num_tasks; i++) {
        remaining_time[i] = tasks[i].execution_time;
    }
    printf("Scheduling:\n");
    while (1) {
        int earliest_deadline_task = -1;
        int earliest_deadline = 999;
        for (int i = 0; i < num_tasks; i++) {
            if (remaining_time[i] > 0 && tasks[i].deadline < earliest_deadline) {
                earliest_deadline = tasks[i].deadline;
                earliest_deadline_task = i;
            }
        }
        if (earliest_deadline_task == -1) {
            break;
        }
        printf("At time %d: Task %d executed\n", current_time,
tasks[earliest_deadline_task].task_id);
        remaining_time[earliest_deadline_task]--;
        current_time++;
        if (remaining_time[earliest_deadline_task] == 0) {
            printf("Task %d completed.\n", tasks[earliest_deadline_task].task_id);
        }
    }
}
int main() {
    int num_tasks;
    printf("Enter number of tasks: ");
    scanf("%d", &num_tasks);
    Task tasks[MAX_TASKS];
    printf("Enter deadline and execution time for each task:\n");
```

```
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d:\n", i + 1);
        printf("Deadline: ");
        scanf("%d", & tasks[i].deadline);
        printf("Execution time: ");
        scanf("%d", & tasks[i].execution_time);
        tasks[i].task_id = i + 1;
    }
    edf_schedule(tasks, num_tasks);
    return 0;
}
```

"C:\Users\yp671\OneDrive\Desktop\opetrating system\earliestdeadlinefirst.exe"

```
Enter number of tasks: 3
Enter deadline and execution time for each task:
Task 1:
Deadline: 7
Execution time: 3
Task 2:
Deadline: 4
Execution time: 2
Task 3:
Deadline: 8
Execution time: 2
Scheduling:
At time 0: Task 2 executed
At time 1: Task 2 executed
Task 2 completed.
At time 2: Task 1 executed
At time 3: Task 1 executed
At time 4: Task 1 executed
Task 1 completed.
At time 5: Task 3 executed
At time 6: Task 3 executed
Task 3 completed.

Process returned 0 (0x0)   execution time : 22.659 s
Press any key to continue.
```

C .proportional scheduling

```
#include <stdio.h>
#define MAX_TASKS 10
typedef struct {
   int weight;
```

```
    int executed_time;
    int task_id;
} Task;
void proportional_schedule(Task tasks[], int num_tasks, int total_time) {
    int remaining_time[MAX_TASKS];
    for (int i = 0; i < num_tasks; i++) {
        remaining_time[i] = tasks[i].weight * total_time;
    }
    printf("Scheduling:\n");
    for (int time = 0; time < total_time; time++) {
        int max_weight_task = -1;
        float max_weight = -1.0;
        for (int i = 0; i < num_tasks; i++) {
            if (remaining_time[i] > 0 && (float)tasks[i].weight / remaining_time[i] > max_weight) {
                max_weight = (float)tasks[i].weight / remaining_time[i];
                max_weight_task = i;
            }
        }
        if (max_weight_task == -1) {
            break;
        }
        printf("At time %d: Task %d executed\n", time, tasks[max_weight_task].task_id);
        remaining_time[max_weight_task]--;
        if (remaining_time[max_weight_task] == 0) {
            printf("Task %d completed.\n", tasks[max_weight_task].task_id);
        }
    }
}
int main() {
    int num_tasks;
    printf("Enter number of tasks: ");
    scanf("%d", &num_tasks);
    Task tasks[MAX_TASKS];
    int total_time;
    printf("Enter total time for scheduling: ");
    scanf("%d", &total_time);
    printf("Enter weight for each task:\n");
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d: ", i + 1);
        scanf("%d", & tasks[i].weight);
        tasks[i].executed_time = 0;
        tasks[i].task_id = i + 1;
    }
    proportional_schedule(tasks, num_tasks, total_time);
```

```
    return 0;
}
```

"C:\Users\yp671\OneDrive\Desktop\opetrating system\propotionalscheduling

```
Enter number of tasks: 3
Enter total time for scheduling: 10
Enter weight for each task:
Task 1: 3
Task 2: 2
Task 3: 1
Scheduling:
At time 0: Task 1 executed
At time 1: Task 1 executed
At time 2: Task 1 executed
At time 3: Task 1 executed
At time 4: Task 1 executed
At time 5: Task 1 executed
At time 6: Task 1 executed
At time 7: Task 1 executed
At time 8: Task 1 executed
At time 9: Task 1 executed

Process returned 0 (0x0)   execution time : 10.442 s
Press any key to continue.
```