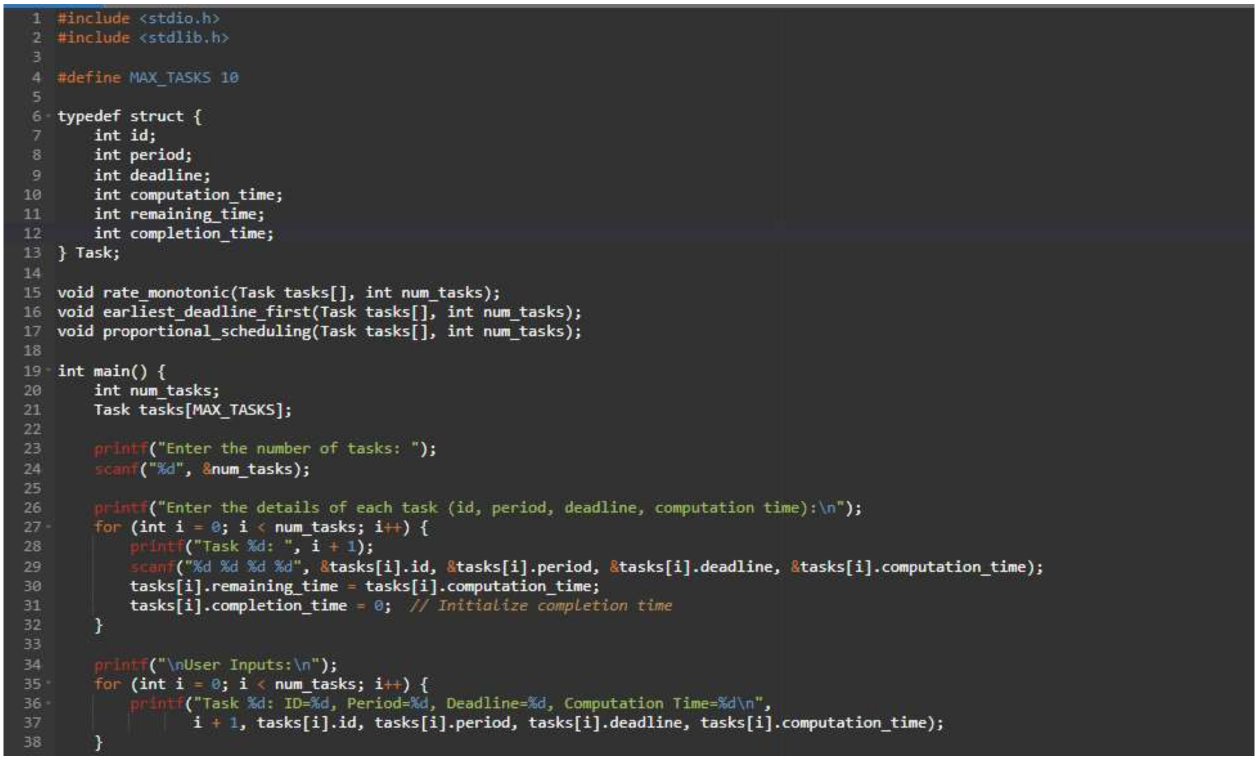
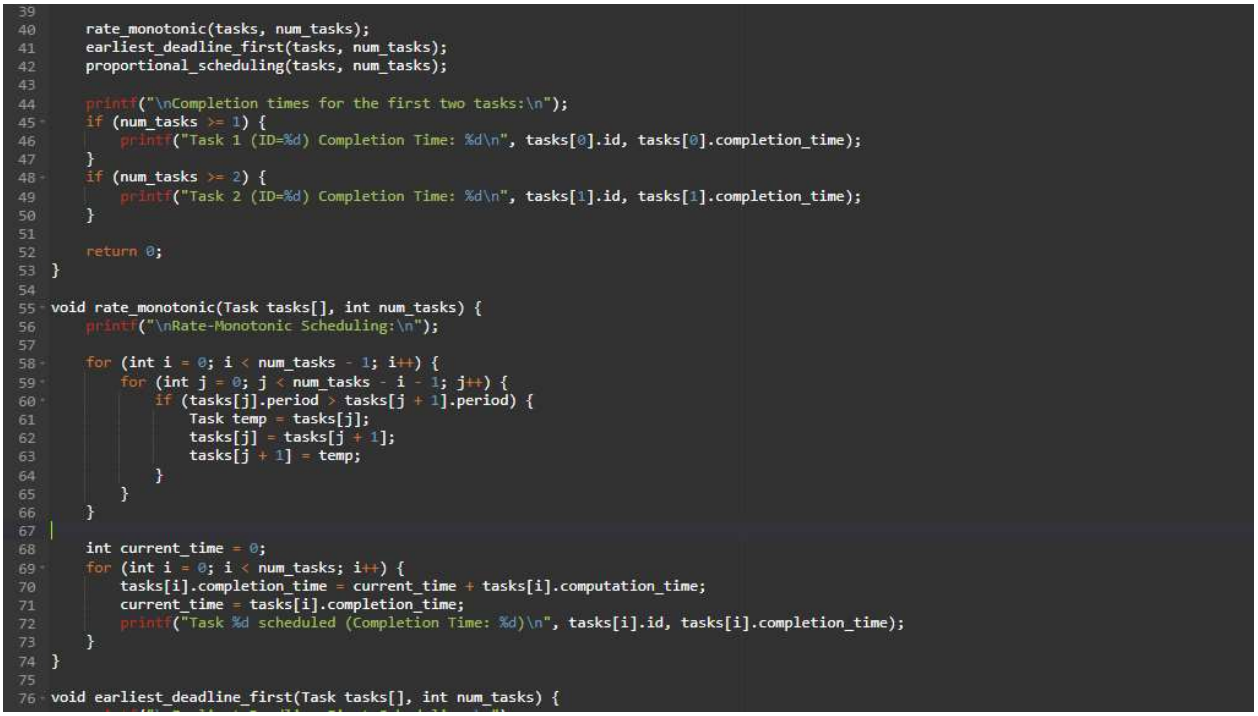
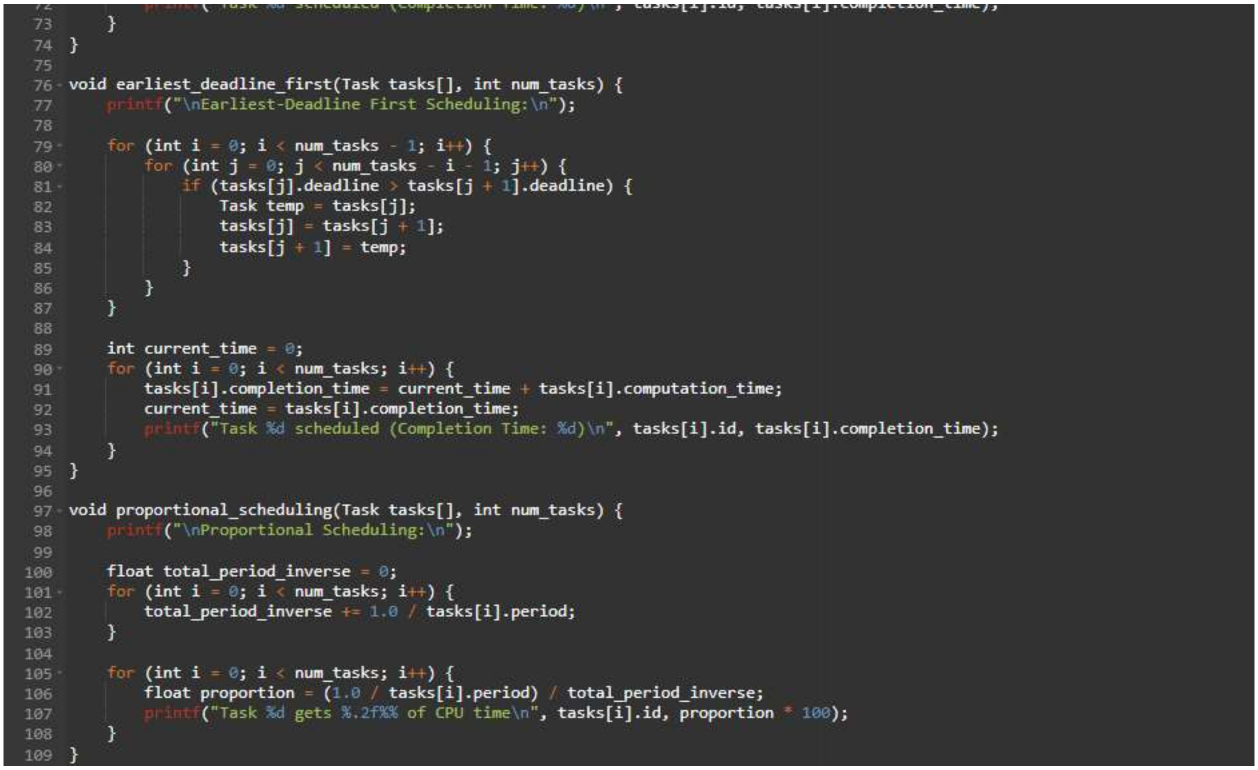
Write a C program to simulate Real-Time CPU Scheduling algorithms:
a) Rate- Monotonic
b) Earliest-deadline First
c) Proportional scheduling
Input:



```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;
    int completion_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;
        tasks[i].completion_time = 0;  // Initialize completion time
    }

    printf("\nUser Inputs:\n");
    for (int i = 0; i < num_tasks; i++) {
        printf("Task %d: ID=%d, Period=%d, Deadline=%d, Computation Time=%d\n",
               i + 1, tasks[i].id, tasks[i].period, tasks[i].deadline, tasks[i].computation_time);
    }
```



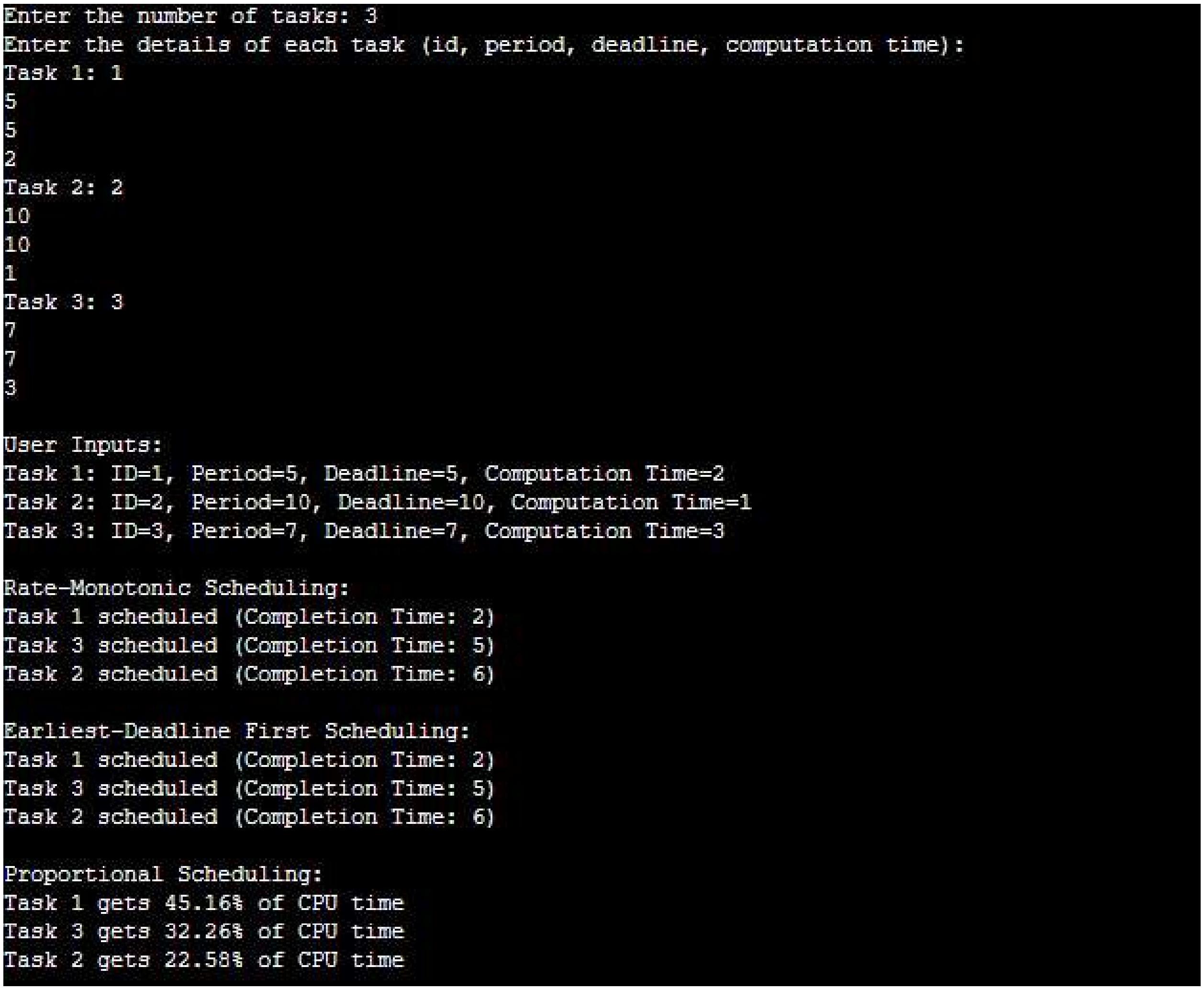
```c

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

    printf("\nCompletion times for the first two tasks:\n");
    if (num_tasks >= 1) {
        printf("Task 1 (ID=%d) Completion Time: %d\n", tasks[0].id, tasks[0].completion_time);
    }
    if (num_tasks >= 2) {
        printf("Task 2 (ID=%d) Completion Time: %d\n", tasks[1].id, tasks[1].completion_time);
    }

    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;
            }
        }
    }

    int current_time = 0;
    for (int i = 0; i < num_tasks; i++) {
        tasks[i].completion_time = current_time + tasks[i].computation_time;
        current_time = tasks[i].completion_time;
        printf("Task %d scheduled (Completion Time: %d)\n", tasks[i].id, tasks[i].completion_time);
    }
}

void earliest_deadline_first(Task tasks[], int num_tasks) {
```



```c
    }
}

void earliest_deadline_first(Task tasks[], int num_tasks) {
```

```c
    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;
            }
        }
    }

    int current_time = 0;
    for (int i = 0; i < num_tasks; i++) {
        tasks[i].completion_time = current_time + tasks[i].computation_time;
        current_time = tasks[i].completion_time;
        printf("Task %d scheduled (Completion Time: %d)\n", tasks[i].id, tasks[i].completion_time);
    }
}

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
1
Task 3: 3
7
7
3

User Inputs:
Task 1: ID=1, Period=5, Deadline=5, Computation Time=2
Task 2: ID=2, Period=10, Deadline=10, Computation Time=1
Task 3: ID=3, Period=7, Deadline=7, Computation Time=3

Rate-Monotonic Scheduling:
Task 1 scheduled (Completion Time: 2)
Task 3 scheduled (Completion Time: 5)
Task 2 scheduled (Completion Time: 6)

Earliest-Deadline First Scheduling:
Task 1 scheduled (Completion Time: 2)
Task 3 scheduled (Completion Time: 5)
Task 2 scheduled (Completion Time: 6)

Proportional Scheduling:
Task 1 gets 45.16% of CPU time
Task 3 gets 32.26% of CPU time
Task 2 gets 22.58% of CPU time
```