g. Earliest deadline first

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

typedef struct {
    int taskId;
    int arrivalTime;
    int executionTime;
    int deadline;
    int remainingTime;
} Task;

int compareDeadline(const void *a, const void *b) {
    Task *task1 = (Task *)a;
    Task *task2 = (Task *)b;
    return task1->deadline - task2->deadline;
}

void performEDF(Task taskList[], int totalTasks) {
    int currentTime = 0;
    int tasksCompleted = 0;

    qsort(taskList, totalTasks, sizeof(Task), compareDeadline);

    printf("Scheduling Order based on Earliest Deadline First (EDF):\n");

    while (tasksCompleted < totalTasks) {
        for (int i = 0; i < totalTasks; i++) {
            if (taskList[i].arrivalTime <= currentTime && taskList[i].remainingTime > 0) {
                printf("At time %d ms: Task %d is executing (Deadline: %d ms)\n", currentTime, taskList[i].t
                taskList[i].remainingTime--;
                if (taskList[i].remainingTime == 0) {
                    tasksCompleted++;
                }
                currentTime++;
                break;
            }
        }
    }
}

int main() {
```

```
    int totalTasks;

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

    Task taskList[totalTasks];

    for (int i = 0; i < totalTasks; i++) {
        taskList[i].taskId = i + 1;
        printf("\nEnter details for Task %d:\n", i + 1);
        printf("Arrival Time: ");
        scanf("%d", &taskList[i].arrivalTime);
        printf("Execution Time: ");
        scanf("%d", &taskList[i].executionTime);
        printf("Deadline: ");
        scanf("%d", &taskList[i].deadline);
        taskList[i].remainingTime = taskList[i].executionTime;
    }

    performEDF(taskList, totalTasks);

    return 0;
}
```

```
Enter the total number of tasks: 3

Enter details for Task 1:
Arrival Time: 0
Execution Time: 3
Deadline: 5

Enter details for Task 2:
Arrival Time: 1
Execution Time: 2
Deadline: 6

Enter details for Task 3:
Arrival Time: 2
Execution Time: 1
Deadline: 4

Scheduling Order based on Earliest Deadline First (EDF):
At time 0 ms: Task 1 is executing (Deadline: 5 ms)
At time 1 ms: Task 2 is executing (Deadline: 6 ms)
```

```
At time 2 ms: Task 2 is executing (Deadline: 6 ms)
At time 3 ms: Task 1 is executing (Deadline: 5 ms)
At time 4 ms: Task 1 is executing (Deadline: 5 ms)
```