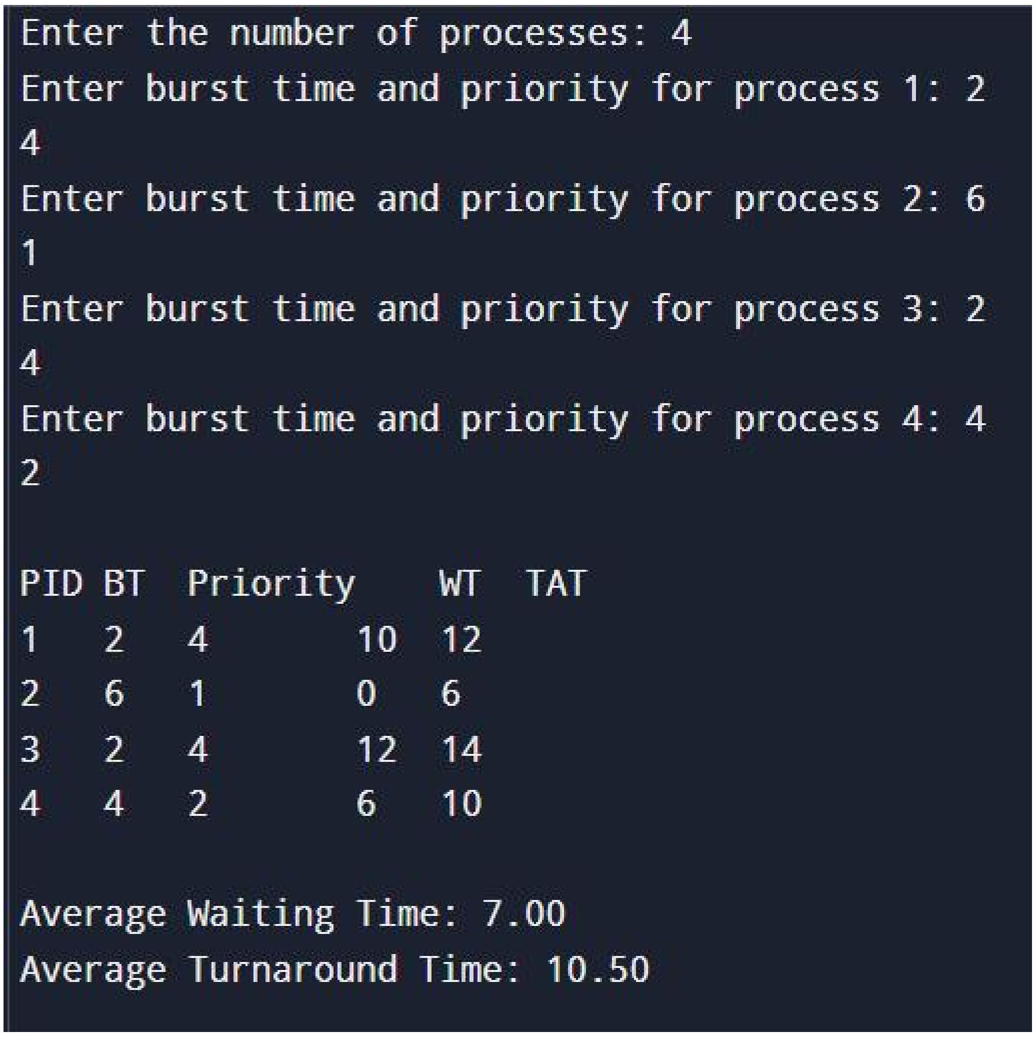
6. Construct a C program to implement preemptive priority scheduling algorithm .

Program:

```
#include <stdio.h>
struct Process {
    int pid,burst_time,remaining_time,priority,waiting_time,turnaround_time,is_completed;
};
void preemptive_priority_scheduling(struct Process processes[], int n) {
    int time = 0, completed = 0, min_priority = -1, current_process = -1;
    int total_waiting_time = 0, total_turnaround_time = 0;
    while (completed != n) {
        min_priority = -1;
        current_process = -1;
        for (int i = 0; i < n; i++) {
            if (processes[i].remaining_time > 0 &&
                (min_priority == -1 || processes[i].priority < min_priority)) {
                min_priority = processes[i].priority;
                current_process = i;
            }}
        if (current_process == -1) break;
        processes[current_process].remaining_time--;
        time++;
        if (processes[current_process].remaining_time == 0) {
            processes[current_process].is_completed = 1;
            completed++;
            processes[current_process].turnaround_time = time;
            processes[current_process].waiting_time = processes[current_process].turnaround_time -
processes[current_process].burst_time;
            total_waiting_time += processes[current_process].waiting_time;
            total_turnaround_time += processes[current_process].turnaround_time;
        }}
    printf("\nPID\tBT\tPriority\tWT\tTAT\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t%d\t%d\t\t%d\t%d\n", processes[i].pid, processes[i].burst_time,
               processes[i].priority, processes[i].waiting_time, processes[i].turnaround_time);
    }
    printf("\nAverage Waiting Time: %.2f", (float)total_waiting_time / n);
    printf("\nAverage Turnaround Time: %.2f\n", (float)total_turnaround_time / n);
}
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process processes[n];
    for (int i = 0; i < n; i++) {
        processes[i].pid = i + 1;
        printf("Enter burst time and priority for process %d: ", i + 1);
        scanf("%d%d", &processes[i].burst_time, &processes[i].priority);
        processes[i].remaining_time = processes[i].burst_time;
        processes[i].is_completed = 0;
    }
    preemptive_priority_scheduling(processes, n);
    return 0;
}
```

Output:



```
Enter the number of processes: 4
Enter burst time and priority for process 1: 2
4
Enter burst time and priority for process 2: 6
1
Enter burst time and priority for process 3: 2
4
Enter burst time and priority for process 4: 4
2

PID BT  Priority    WT  TAT
1   2   4       10  12
2   6   1       0   6
3   2   4       12  14
4   4   2       6   10

Average Waiting Time: 7.00
Average Turnaround Time: 10.50
```