```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
struct Process
{
   int pid;      // process ID
   int bt;       // CPU burst time
   int priority; // priority of the process
   int finish;   // finish time
   int turnaround; // turnaround time
   int waiting;   // waiting time
};

bool sortProcesses(struct Process a, struct Process b)
{
   return (a.priority > b.priority);
}

void findWaitingTurnaroundTime(struct Process proc[], int n)
{
   // Sort processes based on priority
   qsort(proc, n, sizeof(struct Process), (const void *)sortProcesses);

   // Calculate finish time, turnaround time, and waiting time
   proc[0].finish = proc[0].bt;
   proc[0].turnaround = proc[0].finish;
   proc[0].waiting = 0;

   for (int i = 1; i < n; i++)
   {
      proc[i].finish = proc[i - 1].finish + proc[i].bt;
```

```
        proc[i].turnaround = proc[i].finish;
        proc[i].waiting = proc[i - 1].finish;
    }
}

void displayResults(struct Process proc[], int n)
{
    printf("\nProcesses  Burst time  Priority  Finish time  Turnaround time  Waiting time\n");

    for (int i = 0; i < n; i++)
    {
        printf("   %d\t\t%d\t    %d\t\t%d\t\t%d\t\t%d\n", proc[i].pid, proc[i].bt, proc[i].priority,
proc[i].finish, proc[i].turnaround, proc[i].waiting);
    }
}

void calculateAverages(struct Process proc[], int n)
{
    int total_waiting = 0, total_turnaround = 0;

    for (int i = 0; i < n; i++)
    {
        total_waiting += proc[i].waiting;
        total_turnaround += proc[i].turnaround;
    }

    float avg_waiting = (float)total_waiting / (float)n;
    float avg_turnaround = (float)total_turnaround / (float)n;

    printf("\nAverage Waiting Time = %f", avg_waiting);
    printf("\nAverage Turnaround Time = %f", avg_turnaround);
```

```
}

int main()
{
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);

    struct Process *proc = (struct Process *)malloc(n * sizeof(struct Process));

    printf("Enter process details (Process ID, Burst time, Priority):\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d %d %d", &proc[i].pid, &proc[i].bt, &proc[i].priority);
        proc[i].finish = 0;
        proc[i].turnaround = 0;
        proc[i].waiting = 0;
    }

    findWaitingTurnaroundTime(proc, n);
    displayResults(proc, n);
    calculateAverages(proc, n);

    free(proc);
    return 0;
}
```

Output-

```
Enter the number of processes: 3
Enter process details (Process ID, Burst time, Priority):
1 10 1
2 8 3
3 5 2

Processes  Burst time  Priority  Finish time  Turnaround time  Waiting time
   1           10          1          10              10              0
   2           8           3          18              18              10
   3           5           2          23              23              18
```