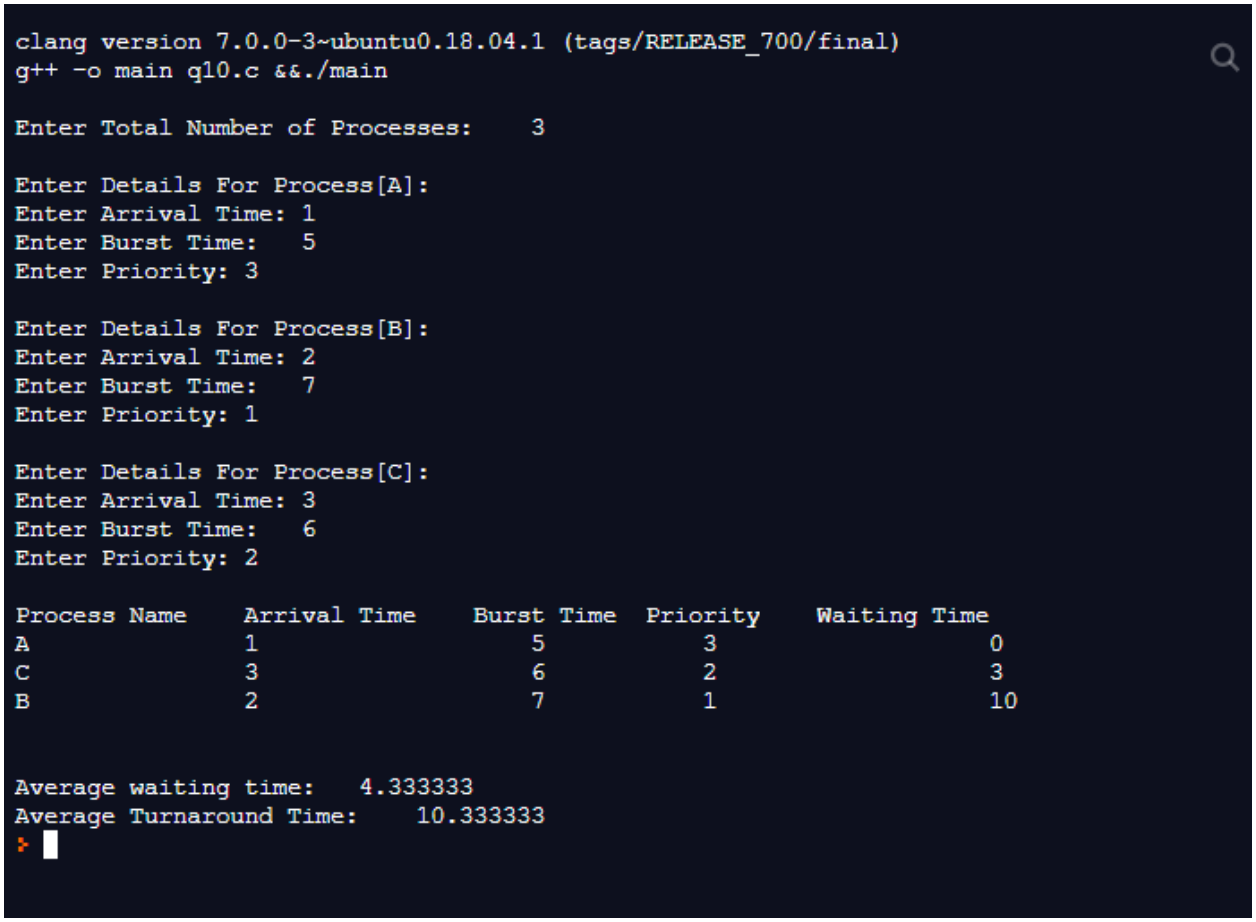
## Ques 10: Write a program to implement preemptive priority based scheduling algorithm

```
#include<stdio.h>

struct process
{
    char process_name;
    int at, bt, ct, wt, tat, prior;
    int status;
}p_queue[10];

int limit;

void Arrival_Time_Sorting()
{
    struct process temp;
    int i, j;
    for(i = 0; i < limit - 1; i++)
    {
        for(j = i + 1; j < limit; j++)
        {
            if(p_queue[i].at > p_queue[j].at)
            {
                temp = p_queue[i];
                p_queue[i] = p_queue[j];
                p_queue[j] = temp;
            }
        }
    }
}

 int main()
{
    int i, time = 0, bt = 0, largest;
    char c;
    float wait_time = 0, tat = 0, awt, atat;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    for(i = 0, c = 'A'; i < limit; i++, c++)
```

```
    {
        p_queue[i].process_name = c;
        printf("\nEnter Details For Process[%C]:\n",p_queue[i].process_name);
        printf("Enter Arrival Time:\t");
        scanf("%d", &p_queue[i].at );
        printf("Enter Burst Time:\t");
        scanf("%d", &p_queue[i].bt);
        printf("Enter Priority:\t");
        scanf("%d", &p_queue[i].prior);
        p_queue[i].status = 0;
        bt = bt + p_queue[i].bt;
    }
    Arrival_Time_Sorting();
    p_queue[9].prior = -9999;
    printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");
    for(time = p_queue[0].at; time < bt;)
    {
        largest = 9;
        for(i = 0; i < limit; i++)
        {
            if(p_queue[i].at <= time && p_queue[i].status != 1 && p_queue[i].prior >
p_queue[largest].prior)
            {
                largest = i;
            }
        }
        time = time + p_queue[largest].bt;
        p_queue[largest].ct = time;
        p_queue[largest].wt = p_queue[largest].ct - p_queue[largest].at -
p_queue[largest].bt;
        p_queue[largest].tat = p_queue[largest].ct - p_queue[largest].at;
        p_queue[largest].status = 1;
        wait_time = wait_time + p_queue[largest].wt;
        tat = tat + p_queue[largest].tat;
        printf("\n%c\t\t\t\t%d\t\t\t\t\t%d\t\t\t%d\t\t\t\t\t%d",
p_queue[largest].process_name, p_queue[largest].at, p_queue[largest].bt,
p_queue[largest].prior, p_queue[largest].wt);
    }
    awt = wait_time / limit;
    atat = tat / limit;
    printf("\n\n\nAverage waiting time:\t%f\n", awt);
    printf("Average Turnaround Time:\t%f\n", atat);
```

}

OUTPUT:


clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
g++ -o main q10.c &&./main
Enter Total Number of Processes: 3
Enter Details For Process[A]:
Enter Arrival Time: 1
Enter Burst Time: 5
Enter Priority: 3
Enter Details For Process[B]:
Enter Arrival Time: 2
Enter Burst Time: 7
Enter Priority: 1
Enter Details For Process[C]:
Enter Arrival Time: 3
Enter Burst Time: 6
Enter Priority: 2
Process Name Arrival Time Burst Time Priority Waiting Time
A 1 5 3 0
C 3 6 2 3
B 2 7 1 10
Average waiting time: 4.333333
Average Turnaround Time: 10.333333