```
#include<stdio.h>

struct process
{
    char process_name;
    int arrival_time,burst_time,ct, wait_time, turnaround_time, priority;
    int status;
}pro_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(pro_queue[i].arrival_time > pro_queue[j].arrival_time)
            {
                temp = pro_queue[i];
                pro_queue[i] = pro_queue[j];
                pro_queue[j] = temp;
            }
        }
    }
}
```

```
int main()
{
    int i,time = 0,burst_time = 0,max;
    char c;
    float wait_time = 0, turnaround_time = 0, avg_waitTime, avg_turnTime;

    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    for(i=0,c ='A';i<limit;i++,c++)
    {
        pro_queue[i].process_name = c;
        printf("\nEnter Details For Process[%C]:\n", pro_queue[i].process_name);
        printf("Enter Arrival Time:\t");
        scanf("%d", &pro_queue[i].arrival_time );
        printf("Enter Burst Time:\t");
        scanf("%d", &pro_queue[i].burst_time);
        printf("Enter Priority:\t");
        scanf("%d", &pro_queue[i].priority);
        pro_queue[i].status = 0;
        burst_time = burst_time + pro_queue[i].burst_time;
    }
    arrival_time_sorting();

    printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");
    for(time=pro_queue[0].arrival_time;time<burst_time;)
    {
        max = 9;
        for(i =0;i<limit;i++)
        {
```

```
            if(pro_queue[i].arrival_time<=time && pro_queue[i].status != 1 &&
pro_queue[i].priority>pro_queue[max].priority)
            {
                max = i;
                pro_queue[i].priority=pro_queue[i].priority+1;
            }
            else
            {
                    pro_queue[i].priority=pro_queue[i].priority+2;
                                        }
        }
        time=time+pro_queue[max].burst_time;
        pro_queue[max].ct = time;
        pro_queue[max].wait_time = pro_queue[max].ct - pro_queue[max].arrival_time -
pro_queue[max].burst_time;
        pro_queue[max].turnaround_time = pro_queue[max].ct - pro_queue[max].arrival_time;
        pro_queue[max].status = 1;
        wait_time = wait_time + pro_queue[max].wait_time;
        turnaround_time = turnaround_time + pro_queue[max].turnaround_time;
        printf("\n %c\t\t%d\t\t%d\t\t%d\t\t%d", pro_queue[max].process_name,
pro_queue[max].arrival_time,pro_queue[max].burst_time,pro_queue[max].priority,
pro_queue[max].wait_time);
    }
    avg_waitTime = wait_time / limit;
    avg_turnTime = turnaround_time / limit;
    printf("\n\nAverage waiting time:\t%f\n",avg_waitTime);
    printf("Average Turnaround Time:\t%f\n",avg_turnTime);
}
```