```
#include<stdio.h>
struct process
{
    char process_name;
    int arrival_time, burst_time, ct, waiting_time, turnaround_time,
priority;
    int status;
}process_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(process_queue[i].arrival_time > process_queue[j].arrival_time)
            {
                temp = process_queue[i];
                process_queue[i] = process_queue[j];
                process_queue[j] = temp;
            }
        }
    }
}
int main()
{
    int i, time = 0, burst_time = 0, largest;
    char c;
    float wait_time = 0, turnaround_time = 0, average_waiting_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    for(i = 0, c = 'A'; i < limit; i++, c++)
    {
```

```
        process_queue[i].process_name = c;
        printf("\nEnter Details For Process[%C]:\n",
process_queue[i].process_name);
        printf("Enter Arrival Time:\t");
        scanf("%d", &process_queue[i].arrival_time );
        printf("Enter Burst Time:\t");
        scanf("%d", &process_queue[i].burst_time);
        printf("Enter Priority:\t");
        scanf("%d", &process_queue[i].priority);
        process_queue[i].status = 0;
        burst_time = burst_time + process_queue[i].burst_time;
    }
    Arrival_Time_Sorting();
    process_queue[9].priority = -9999;
    printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting
Time");
    for(time = process_queue[0].arrival_time; time < burst_time;)
    {
        largest = 9;
        for(i = 0; i < limit; i++)
        {
            if(process_queue[i].arrival_time <= time &&
process_queue[i].status != 1 && process_queue[i].priority >
process_queue[largest].priority)
            {
                largest = i;
```

```
            }
        }
        time = time + process_queue[largest].burst_time;
        process_queue[largest].ct = time;
        process_queue[largest].waiting_time = process_queue[largest].ct -
process_queue[largest].arrival_time - process_queue[largest].burst_time;
        process_queue[largest].turnaround_time =
process_queue[largest].ct - process_queue[largest].arrival_time;
        process_queue[largest].status = 1;
        wait_time = wait_time + process_queue[largest].waiting_time;
        turnaround_time = turnaround_time +
process_queue[largest].turnaround_time;
        printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d",
process_queue[largest].process_name, process_queue[largest].arrival_time,
process_queue[largest].burst_time, process_queue[largest].priority,
process_queue[largest].waiting_time);
    }
    average_waiting_time = wait_time / limit;
    average_turnaround_time = turnaround_time / limit;
    printf("\n\nAverage waiting time:\t%f\n", average_waiting_time);
    printf("Average Turnaround Time:\t%f\n",
average_turnaround_time);
}
```