```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
static int n;
struct process
{
        int priority;
        int ServiceTime;
        int ArrivalTime;
        int processedFlag;
        char ProcessID[5];
        process *link;
}*front=NULL,*rear=NULL;
void swap(process *a,process *b)
{
   process *temp = a;
   a=b;
   b=temp;
}
void sortProcess()
{
   process *l=NULL,*p;
   if(front==NULL)
      return ;
   else
   {
        int swaped;
        do
        {
```

```
            swaped=0;
            p=front;
        while(p->link!=NULL)
        {
           if(p->ArrivalTime> p->link->ArrivalTime)
                {
                   swap(p,p->link);
                   swaped=1;
                   }
           p=p->link;
        }
        l=p;
                }while(swaped);
   }
}
void processCreation()
{
        printf("Enter Number of process you want to enter:");
        scanf("%d",&n);
        process *newProcess;
        for(int i=0;i<n;i++)
        {
                newProcess=(struct process*)malloc(sizeof(process));
                printf("\nEnter Process ID:");
                scanf("%s",&newProcess->ProcessID);
                printf("Enter Process Arival time:");
                scanf("%d",&newProcess->ArrivalTime);
                printf("Enter Process Service Time:");
                scanf("%d",&newProcess->ServiceTime);
```

```
                newProcess->priority=0;
                newProcess->processedFlag=0;
                newProcess->link=NULL;
                if(front==NULL)
                        rear=front=newProcess;
                else
                {
                        rear->link=newProcess;
                        rear=newProcess;
                }
        }
        sortProcess();
}
void processor()
{
        process *p=front,readyQueue[n];
        readyQueue[0]=*p;
        for(int i=p->ArrivalTime,k=0;i<rear->ArrivalTime;i++)
        {
                readyQueue[k].processedFlag=1;
                readyQueue[k].ServiceTime--;
                readyQueue[k].priority++;
                p=p->link;
                if(i+1==p->ArrivalTime)
                {
                        if(readyQueue[k].ServiceTime==0)
                        {
                                printf("%s is executed with time
%d",readyQueue[k].ProcessID,i);
```

```
                        readyQueue[k]=*p;
                }
                else
                {
                        if((p->processedFlag==0)||(p->priority>readyQueue[k].priority))
                        {
                                readyQueue[k+1]=readyQueue[k];
                                readyQueue[k]=*p;
                                k++;
                        }
                        else
                        {
                                readyQueue[k+1]=*p;
                                k++;
                        }
                }
        }
    }
}
int main()
{

    printf("~~~~~~~~~~~~~~~Welcome To Process Scheduler~~~~~~~~~~~~~~~~\a\n");
    printf("\n\n~~~~~~~~~~~~~~~~~~~Priority
Scheduling~~~~~~~~~~~~~~~~~~~~\a\n");
    getch();
    processCreation();
    processor();
}
```