#include <stdio.h>   
#include <stdlib.h>    
#include <string.h>    
typedef struct node    
{    
   char name[10];  /\*进程标识符\*/    
   int prio;   /\*进程优先数\*/    
   int round;  /\*进程时间轮转时间片\*/    
   int cputime; /\*进程占用CPU时间\*/    
   int needtime; /\*进程到完成还要的时间\*/    
   int count;  /\*计数器\*/    
   char state; /\*进程的状态\*/    
   struct node \*next; /\*链指针\*/    
}PCB;    
PCB \*finish,\*ready,\*tail,\*run; /\*队列指针\*/    
int N; /\*进程数\*/    
/\*将就绪队列中的第一个进程投入运行\*/    
firstin()    
{    
   run=ready;   /\*就绪队列头指针赋值给运行头指针\*/    
   run->state=&apos;R&apos;;   /\*进程状态变为运行态\*/    
   ready=ready->next;  /\*就绪对列头指针后移到下一进程\*/    
}    
/\*标题输出函数\*/    
void prt1(char a)    
{    
   if(toupper(a)==&apos;P&apos;) /\*优先数法\*/    
      printf("  进程号   cpu时间  所需时间  优先数    状态\n");    
   else    
      printf("  进程号   cpu时间  所需时间   记数   时间片       状态\n");    
}    
/\*进程PCB输出\*/    
void prt2(char a,PCB \*q)    
{    
   if(toupper(a)==&apos;P&apos;)  /\*优先数法的输出\*/    
      printf("  %-10s%-10d%-10d%-10d %c\n",q->name,    
       q->cputime,q->needtime,q->prio,q->state);    
   else/\*轮转法的输出\*/    
      printf("  %-10s%-10d%-10d%-10d%-10d %-c\n",q->name,    
       q->cputime,q->needtime,q->count,q->round,q->state);    
}    
/\*输出函数\*/    
void prt(char algo)    
{    
   PCB \*p;    
   prt1(algo);  /\*输出标题\*/    
   if(run!=NULL) /\*如果运行指针不空\*/    
      prt2(algo,run); /\*输出当前正在运行的PCB\*/    
   p=ready;  /\*输出就绪队列PCB\*/    
   while(p!=NULL)    
   {    
      prt2(algo,p);    
      p=p->next;    
   }    
   p=finish;  /\*输出完成队列的PCB\*/    
   while(p!=NULL)    
   {    
      prt2(algo,p);    
      p=p->next;    
   }    
   getchar();  /\*压任意键继续\*/    
}    
/\*优先数的插入算法\*/    
insert1(PCB \*q)    
{    
   PCB \*p1,\*s,\*r;    
   int b;    
   s=q;  /\*待插入的PCB指针\*/    
   p1=ready; /\*就绪队列头指针\*/    
   r=p1; /\*r做p1的前驱指针\*/    
   b=1;    
   while((p1!=NULL)&&b)  /\*根据优先数确定插入位置\*/    
      if(p1->prio>=s->prio)    
      {    
     r=p1;    
     p1=p1->next;    
      }    
      else    
     b=0;    
   if(r!=p1)  /\*如果条件成立说明插入在r与p1之间\*/    
   {    
      r->next=s;    
      s->next=p1;    
   }    
   else    
   {    
      s->next=p1;  /\*否则插入在就绪队列的头\*/    
      ready=s;    
   }    
}    
/\*优先数创建初始PCB信息\*/    
void create(char alg)    
{    
   PCB \*p;    
   int i,time;    
   char na[10];    
   ready=NULL; /\*就绪队列头指针\*/    
   finish=NULL;  /\*完成队列头指针\*/    
   run=NULL; /\*运行队列指针\*/    
   printf("输入进程号和运行时间：\n"); /\*输入进程标识和所需时间创建PCB\*/    
   for(i=1;i<=N;i++)    
   {    
      p=(PCB \*)malloc(sizeof(PCB));    
      scanf("%s",na);    
      scanf("%d",&time);    
      strcpy(p->name,na);    
      p->cputime=0;    
      p->needtime=time;    
      p->state=&apos;w&apos;;    
      p->prio=50-time;    
      if(ready!=NULL) /\*就绪队列不空调用插入函数插入\*/    
     insert1(p);    
      else    
      {    
     p->next=ready; /\*创建就绪队列的第一个PCB\*/    
     ready=p;    
      }    
   }    
   printf("          优先数算法输出信息:\n");    
   printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");    
   prt(alg);  /\*输出进程PCB信息\*/    
   run=ready; /\*将就绪队列的第一个进程投入运行\*/    
   ready=ready->next;    
   run->state=&apos;R&apos;;    
}    
/\*优先数调度算法\*/    
priority(char alg)    
{    
   while(run!=NULL)  /\*当运行队列不空时，有进程正在运行\*/    
   {    
      run->cputime=run->cputime+1;    
      run->needtime=run->needtime-1;    
      run->prio=run->prio-3; /\*每运行一次优先数降低3个单位\*/    
      PCB \*p;  
      p=ready;  
      while(p!=NULL)    
      {    
          p->prio=p->prio+1; /\*每等待一次优先数升高1个单位\*/      
          p=p->next;    
      }    
      if(run->needtime==0)  /\*如所需时间为0将其插入完成队列\*/    
      {    
         run->next=finish;    
         finish=run;    
         run->state=&apos;F&apos;;  /\*置状态为完成态\*/    
         run=NULL;  /\*运行队列头指针为空\*/    
         if(ready!=NULL) /\*如就绪队列不空\*/   
         {    
            firstin(); /\*将就绪对列的第一个进程投入运行\*/    
         }  
      }    
      else /\*没有运行完同时优先数不是最大，则将其变为就绪态插入到就绪队列\*/    
     if((ready!=NULL)&&(run->prio<ready->prio))    
     {    
        run->state=&apos;W&apos;;    
        insert1(run);    
        firstin(); /\*将就绪队列的第一个进程投入运行\*/    
     }    
      prt(alg); /\*输出进程PCB信息\*/    
   }    
}    
/\*主函数\*/    
main()    
{    
   char algo;  /\*算法标记\*/    
   printf("输入P确定算法：优先数算法\n");    
   scanf("%c",&algo); /\*输入字符确定算法\*/    
   printf("输入进程数：\n");    
   scanf("%d",&N); /\*输入进程数\*/    
   if(algo==&apos;P&apos;||algo==&apos;p&apos;)    
   {    
      create(algo); /\*优先数法\*/    
      priority(algo);    
   }   
}