**C**P**U SCHEDULING** P**ROGRAMS**

**Sjf**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | #include<stdio.h>  #include<stdlib.h>  int i=0,n;  struct algorithm{  int at,bt,ct,tt,wt,id;};  typedef struct algorithm algo;  algo p[10];  void swap(int \*b,int \*c)  {      int tem;      tem=\*c;      \*c=\*b;      \*b=tem;  }  void main()  {      int flag=0;      float avgtt=0,avgwt=0;      printf("enter no of process");      scanf("%d",&n);      printf("\n enter the arrival time and burst time\n");      for(i=0;i<n;i++)      {          p[i].id=i+1;          printf("\n p %d ->",i+1);          scanf("%d%d",&p[i].at,&p[i].bt);      }      int sum=0;      for(int i=0;i<n;i++)          {              for(int j=0;j<n-i-1;j++)              {                  if(p[j].at > p[j+1].at)                  {                        swap(&p[j+1].id,&p[j].id);                        swap(&p[j+1].at,&p[j].at);                        swap(&p[j+1].bt,&p[j].bt);                  }              }          }      int cmp=0;      i=0;          int y=p[0].at;      for(int k=0;k<n;k++)          {              int min=p[k].bt;              for(int j=k+1;j<n;j++)              {                  if(min>p[j].bt && p[j].at<=y)                  {                        min=p[j].bt;                        swap(&p[k].id,&p[j].id);                        swap(&p[k].at,&p[j].at);                        swap(&p[k].bt,&p[j].bt);                  }              }              if(p[k].at>cmp)                      cmp=cmp+p[k].bt+(p[k].at-cmp);                  else                      cmp=cmp+p[k].bt;                  p[k].ct=cmp;                  y=cmp;          }      printf("\n ID\t AT\t BT\t CT\t TT\t WT");      for(i=0;i<n;i++)      {          p[i].tt=p[i].ct - p[i].at;          p[i].wt=p[i].tt - p[i].bt;          avgtt+=p[i].tt;          avgwt+=p[i].wt;          printf("\n %d\t %d\t %d\t %d\t %d\t %d",p[i].id,p[i].at,p[i].bt,p[i].ct,p[i].tt,p[i].wt);      }      printf("\n avgtt=%f",(avgtt/n));      printf("\n avgwt=%f",(avgwt/n));      printf("\n");      cmp=0;      for(int i=0;i<n;i++)      {          if(p[i].at<=cmp)         {             printf("|P%d(%d)",p[i].id,p[i].bt);              cmp+=p[i].bt;          }          else          {              printf("|XX(%d)",p[i].at-cmp);              cmp=p[i].at;              i--;          }      }      printf("|\n0 ");      cmp=0;      for(int i=0;i<n;i++)      {          if(p[i].at<=cmp)          {              printf("%d ",p[i].ct);              cmp+=p[i].bt;          }          else{              printf("%d ",p[i].at);              cmp=p[i].at;              i--;          }      }  }  **Round Robin**   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | #include<stdio.h>  #include<stdlib.h>  int i=0,n;  struct algorithm{  int at,bt,ct,tt,wt,id,dt;};  typedef struct algorithm algo;  algo p[10];  void swap(int \*b,int \*c)  {      int tem;      tem=\*c;      \*c=\*b;      \*b=tem;  }  void main()  {      int i,j,k,q;      int rq[20],f=0,r=-1;      float avgtt=0,avgwt=0;      int ids[100],nos[100];      printf("enter no of process");      scanf("%d",&n);      printf("enter time quantum");      scanf("%d",&q);      printf("\n enter the arrival time and burst time\n");      for(i=0;i<n;i++)      {          p[i].id=i+1;          printf("\n p %d ->",i+1);          scanf("%d%d",&p[i].at,&p[i].bt);          p[i].dt=p[i].bt;      }      int sum=0;      for(int i=0;i<n;i++)          {              for(int j=0;j<n-i-1;j++)              {                  if(p[j].at > p[j+1].at)                  {                        swap(&p[j+1].id,&p[j].id);                        swap(&p[j+1].at,&p[j].at);                        swap(&p[j+1].bt,&p[j].bt);                        swap(&p[j+1].dt,&p[j].dt);                  }              }          }      int y=p[n-1].at,x=p[0].at,a=0,z;      int cmp=x;      if(x!=0)      {          ids[a]=-1;          nos[a++]=x;      }      rq[++r]=0;      k=1;      while(cmp<=y)      {          z=rq[f++];          if(p[z].bt<q)          {              cmp+=p[z].bt;              p[z].bt=0;          }          else{              cmp+=q;              p[z].bt-=q;          }          ids[a]=p[z].id;          nos[a++]=cmp;          while(p[k].at<=cmp && k<n)          {              rq[++r]=k;              k++;          }          if(p[z].bt!=0)              rq[++r]=z;          else              p[z].ct=cmp;      }      while(f-r!=1)      {          z=rq[f++];          if(p[z].bt<q)          {              cmp+=p[z].bt;              p[z].bt=0;          }          else{              cmp+=q;              p[z].bt-=q;          }          ids[a]=p[z].id;          nos[a++]=cmp;          if(p[z].bt!=0)              rq[++r]=z;          else              p[z].ct=cmp;      }       printf("\n ID\t AT\t BT\t CT\t TT\t WT");      for(i=0;i<n;i++)      {          p[i].tt=p[i].ct - p[i].at;          p[i].wt=p[i].tt - p[i].dt;          avgtt+=p[i].tt;          avgwt+=p[i].wt;          printf("\n %d\t %d\t %d\t %d\t %d\t %d",p[i].id,p[i].at,p[i].dt,p[i].ct,p[i].tt,p[i].wt);      }      printf("\n avgtt=%f",(avgtt/n));      printf("\n avgwt=%f \n",(avgwt/n));       for(i=0;i<a;i++)     {         if(ids[i]==ids[i+1])          continue;         if(ids[i]==-1)          printf("| XXX \t",ids[i]);         else          printf("| P%d \t",ids[i]);     }      printf(" | \n0 \t");      for(i=0;i<a;i++)     {         if(ids[i]==ids[i+1])          continue;         else          printf("%d \t",nos[i]);     }  }   |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **Priority**   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | #include<stdio.h>  #include<stdlib.h>  int i=0,n;  struct algorithm{  int at,bt,ct,tt,wt,id,pr;};  typedef struct algorithm algo;  algo p[10];  void swap(int \*b,int \*c)  {      int tem;      tem=\*c;      \*c=\*b;      \*b=tem;  }  void main()  {      int flag=0;      float avgtt=0,avgwt=0;      printf("enter no of process");      scanf("%d",&n);      printf("\n enter the arrival time  burst time and priority \n");      for(i=0;i<n;i++)      {          p[i].id=i+1;          printf("\n p %d ->",i+1);          scanf("%d%d%d",&p[i].at,&p[i].bt,&p[i].pr);      }      int sum=0;      for(int i=0;i<n;i++)          {              for(int j=0;j<n-i-1;j++)              {                  if(p[j].at > p[j+1].at)                  {                        swap(&p[j+1].id,&p[j].id);                        swap(&p[j+1].at,&p[j].at);                        swap(&p[j+1].bt,&p[j].bt);                        swap(&p[j+1].pr,&p[j].pr);                  }              }          }      int cmp=0;      i=0;      int y=p[0].at;      for(int k=0;k<n;k++)          {              int min=p[k].pr;              for(int j=k+1;j<n;j++)              {                  if(min>p[j].pr && p[j].at<=y)                  {                        min=p[j].pr;                        swap(&p[k].id,&p[j].id);                        swap(&p[k].at,&p[j].at);                        swap(&p[k].bt,&p[j].bt);                        swap(&p[k].pr,&p[j].pr);                  }              }                   if(p[k].at>cmp)                      cmp=cmp+p[k].bt+(p[k].at-cmp);                  else                      cmp=cmp+p[k].bt;                  p[k].ct=cmp;                  y=cmp;          }      printf("\n ID\t AT\t BT\t PR\t CT\t TT\t WT");      for(i=0;i<n;i++)      {          p[i].tt=p[i].ct - p[i].at;          p[i].wt=p[i].tt - p[i].bt;          avgtt+=p[i].tt;          avgwt+=p[i].wt;          printf("\n %d\t %d\t %d\t %d\t %d\t %d\t %d",p[i].id,p[i].at,p[i].bt,p[i].pr,p[i].ct,p[i].tt,p[i].wt);      }      printf("\n avgtt=%f",(avgtt/n));      printf("\n avgwt=%f\n\n",(avgwt/n));      int cp=0;      for(int i=0;i<n;i++)      {          if(p[i].at<=cp)         {             printf("|P%d(%d)",p[i].id,p[i].bt);              cp+=p[i].bt;          }          else          {              printf("|XX(%d)",p[i].at-cp);              cp=p[i].at;              i--;          }      }      printf("|\n0     ");      cp=0;      for(int i=0;i<n;i++)      {          if(p[i].at<=cp)          {              printf("%d     ",p[i].ct);              cp+=p[i].bt;          }          else{              printf("%d      ",p[i].at);              cp=p[i].at;              i--;          }  }  }  **Priority pre-emptive**   |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | #include<stdio.h>  #include<stdlib.h>  int i=0,n;  struct algorithm{  int at,bt,ct,tt,wt,id,pr,dt;};  typedef struct algorithm algo;  algo p[10];  void swap(int \*b,int \*c)  {      int tem;      tem=\*c;      \*c=\*b;      \*b=tem;  }  void main()  {      int flag=0;      float avgtt=0,avgwt=0;      int ids[100],nos[100];      printf("enter no of process");      scanf("%d",&n);      printf("\n enter the arrival time and burst time and priority\n");      for(i=0;i<n;i++)      {          p[i].id=i+1;          printf("\n p %d ->",i+1);          scanf("%d%d%d",&p[i].at,&p[i].bt,&p[i].pr);          p[i].dt=p[i].bt;      }      int sum=0;      for(int i=0;i<n;i++)          {              for(int j=0;j<n-i-1;j++)              {                  if(p[j].at > p[j+1].at)                  {                        swap(&p[j+1].id,&p[j].id);                        swap(&p[j+1].at,&p[j].at);                        swap(&p[j+1].bt,&p[j].bt);                        swap(&p[j+1].pr,&p[j].pr);                        swap(&p[j+1].dt,&p[j].dt);                  }              }          }      int cmp=0;          int y=p[0].at,x=p[n-1].at,a=0;      for(int k=0;k<n;k++)          {              int min=p[k].pr;              for(int j=k+1;j<n;j++)              {                  if(min>p[j].pr && p[j].at<=y )                  {                        min=p[j].bt;                        swap(&p[k].id,&p[j].id);                        swap(&p[k].at,&p[j].at);                        swap(&p[k].bt,&p[j].bt);                        swap(&p[k].dt,&p[j].dt);                        swap(&p[k].pr,&p[j].pr);                  }              }                   if(p[k].bt==0)                      {                          p[k].ct=cmp;                          continue;}                  if(y<=x)                  {                      if(p[k].at>cmp)                      {                          cmp=cmp+1+(p[k].at-cmp);                          ids[a]=-1;                          nos[a++]=p[k].at;                      }                      else                     {                          cmp++;                          ids[a]=p[k].id;                          nos[a++]=cmp;                     }                      p[k].bt--;                       k--;              }              else{              if(p[k].at>cmp)                      cmp=cmp+p[k].bt+(p[k].at-cmp);                  else                      cmp=cmp+p[k].bt;                  ids[a]=p[k].id;                  nos[a++]=cmp;              p[k].ct=cmp;              }              y=cmp;          }      printf("\n ID\t AT\t BT\t PR\t CT\t TT\t WT");      for(i=0;i<n;i++)      {          p[i].tt=p[i].ct - p[i].at;          p[i].wt=p[i].tt - p[i].dt;          avgtt+=p[i].tt;          avgwt+=p[i].wt;          printf("\n %d\t %d\t %d\t %d\t %d\t %d\t %d",p[i].id,p[i].at,p[i].dt,p[i].pr,p[i].ct,p[i].tt,p[i].wt);      }      printf("\n avgtt=%f",(avgtt/n));      printf("\n avgwt=%f \n",(avgwt/n));      ids[a]=nos[a]=-1;     for(i=0;i<a;i++)     {         if(ids[i]==ids[i+1])          continue;         if(ids[i]==-1)          printf("| XXX \t",ids[i]);         else          printf("| P%d \t",ids[i]);     }      printf(" | \n0 \t");      for(i=0;i<a;i++)     {         if(ids[i]==ids[i+1])          continue;         else          printf("%d \t",nos[i]);     }  }   |  |  | | --- | --- | |  |  | |  |  |  |  | | --- | --- | |  |  | |  | |  | |  |  |  |  | | --- | --- | |  |  | |  |