Operating systems (scheduling programs)

FCFS:

```
#include<stdio.h>
#define M 100
int waitT(int p[],int np,int bt[],int wt[])
{
	wt[0]=0;
	for(int i=1;i<np;i++)
	{
		wt[i]=bt[i-1]+wt[i-1];
	}
}
int trt(int p[],int np,int bt[],int wt[],int tat[])
{
	for(int i=0;i<np;i++)
	{
		tat[i]=bt[i]+wt[i];
	}
}
void avgT(int p[],int np,int bt[])
{
	int wt[np]={0},tat[np]={0};
	float totwt=0,ttat=0;
	waitT(p,np,bt,wt);
	trt(p,np,bt,wt,tat);
	printf("PROCESS\tBURST_TIME\tWAIT_TIME\tTURN_AROUND_TIME\n");
	for(int i=0;i<np;i++)
	{
		totwt=totwt+wt[i];
		ttat=ttat+tat[i];
```

```
            printf("%d\t%d\t\t%d\t\t%d",p[i],bt[i],wt[i],tat[i]);
            printf("\n");
        }
        printf("\nAverage Waiting Time==%f",totwt/np);
        printf("\nAverage Turn Around Time==%f",ttat/np);

}
int main()
{
        int p[M],np,bt[M];
        printf("Enter the NO.OF process\n");
        scanf("%d",&np);
        for(int i=0;i<np;i++)
        {
            printf("Enter the BURST TIME of process P%d\n",i+1);
            p[i]=i+1;
            scanf("%d",&bt[i]);
        }
        printf("\n\n\t\tFIRST COME FIRST SERVE\n\n");
        avgT(p,np,bt);
}
```

SJF:

(NON-PREEMPTIVE):

```
#include<iostream>
using namespace std;
#define M 10
int minbt(int n,int p[],int bt[],int wt[],int t[],int at[],int ct[])
{
	int x;
	float awt=0,atat=t[0];
	for(int i=1;i<n;i++)
	{
		int nop=0;
		for(int j=i;j<n;j++)
		{
			if(at[j]<=ct[i-1])
			{
				nop++;
			}
		}
		int k=i,s=i+nop;
		for(k;k<s;k++)
		{
			for(int l=k+1;l<s;l++)
			{
				if(bt[k]>bt[l])
				{
					x=bt[k];
```

```
                    bt[k]=bt[l];
                    bt[l]=x;

                    x=p[k];
                    p[k]=p[l];
                    p[l]=x;

                    x=at[k];
                    at[k]=at[l];
                    at[l]=x;

                }
            }
        }

        ct[i]=ct[i-1]+bt[i];
        t[i]=ct[i]-at[i];
        wt[i]=t[i]-bt[i];
        /*wt[i]=ct[i-1]-at[i];
        ct[i]=wt[i]+at[i]+bt[i];
        t[i]=bt[i]+wt[i]; */
        awt=awt+wt[i];
        atat=atat+t[i];
    }
    cout<<"\n\n\n";
    cout<<"SCHEDULING PROCESS (S.J.F  NON-PREEMPTIVE)\n\n\n";
    cout<<"PROCESS\t\tARIVAL_TIME\tBURST_TIME\tCOMP_TIME\tT
ART\tWAIT_TIME\t\n";
```

```
	for(int i=0;i<n;i++)
	{
		cout<<"
"<<p[i]<<"\t\t"<<at[i]<<"\t\t"<<bt[i]<<"\t\t"<<ct[i]<<"\t\t"<<t[i]<<"\t"<<wt[i]<
<"\t"<<endl;
	}
	awt/=n;
	atat/=n;
	cout<<"\n\nAVG WAITING TIME=="<<awt<<endl<<"AVG
TRUN_A_TIME=="<<atat<<endl;
}
int main()
{
	int x,n,p[M],bt[M],wt[M],t[M],at[M],ct[M];
	cout<<"ENTER THE NO.OF PROCESS\n";
	cin>>n;
	for(int i=0;i<n;i++)
	{
		cout<<"ENTER THE ARIVAL_TIME AND BURST_TIME FOR
PROCESS "<<(i+1)<<endl;
		cout<<"ARIVAL_TIME::";
		cin>>at[i];
		cout<<"BURST_TIME::";
		cin>>bt[i];
		p[i]=i+1;
	}
	for(int i=0;i<n;i++)
	{
		for(int j=i+1;j<n;j++)
```

```
{
        if(at[i]>at[j])     //BUBBLE SORTING
        {
                x=bt[i];
                bt[i]=bt[j];
                bt[j]=x;

                x=p[i];
                p[i]=p[j];
                p[j]=x;

                x=at[i];
                at[i]=at[j];
                at[j]=x;
        }
        if(at[i]==at[j]) //if times are arival times are equal ... sorting based on burst time.
        {
                if(bt[i]>bt[j])
                {
                        x=bt[i];
                   bt[i]=bt[j];
                   bt[j]=x;

                   x=p[i];
                   p[i]=p[j];
                   p[j]=x;
```

```
                    x=at[i];
                    at[i]=at[j];
                    at[j]=x;

                }
            }
        }
    }
    //wt[0]=0;
    //t[0]=bt[0];
    ct[0]=at[0]+bt[0];
    t[0]=ct[0]-at[0];
    wt[0]=t[0]-bt[0];
    cout<<"ARIVAL_TIME\tPROCESS\tBURST_TIME\n";
    for(int i=0;i<n;i++)
    {
        cout<<" "<<at[i]<<"\t\t"<<p[i]<<"\t"<<bt[i]<<endl;
    }
    minbt(n,p,bt,wt,t,at,ct);
}
```

PREEMPTIVE:

```
#include<iostream>
using namespace std;
#define M 10
int n,p[M],bt[M],wt[M],t[M],at[M],ct[M],tbt[M],st[M],tm=0,i,j;
float pc=0,tat,awt;
int check(int min,int tm);
//int print(int id);
void sjf()
{

        while(pc!=n)
        {
                int min=999;
                min=check(min,tm);
                bt[min]--;
      if(bt[min]==0)
      {
         pc++;
         //print(min);
         st[min]=1;
         ct[min] = tm+1;
         wt[min] = (tm+1) - at[min] - tbt[min];
         t[min] = (tm+1) - at[min];
      }
      tm++;
        }
```

```
}
int check(int min,int tm)
{
        int id=min;
      for(i=0; i<n; i++)
      {
         if(at[i]<=tm && bt[i]<min && st[i]==0 )
         {
               min=bt[i];
               id=i;
                     }
      }
      /*if(min!=id)
      print(id);*/
        return id;
}
/*int print(int id)
{
        cout<<" "<<p[id]<<"\t\t"<<bt[id]<<"\t\t"<<tm<<endl;
}*/
int main()
{

        cout<<"ENTER THE NO.OF PROCESS\n";
        cin>>n;
        for(int i=0;i<n;i++)
        {
```

```
            cout<<"ENTER THE ARIVAL_TIME AND BURST_TIME FOR
PROCESS "<<(i+1)<<endl;
            cout<<"ARIVAL_TIME::";
            cin>>at[i];
            cout<<"BURST_TIME::";
            cin>>bt[i];
            tbt[i]=bt[i];
            p[i]=i+1;
            st[i]=0;
        }
        sjf();
        cout<<"\n\n\t\tSHORTEST REMAINING TIME
SCHEDULING\n\t\t(SJF PREEMPTIVE)\n\n";
        cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time"
<<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-
time"<<endl;
   for(int i=0; i<n; i++)
   {

cout<<"p"<<i+1<<"\t\t"<<tbt[i]<<"\t\t"<<at[i]<<"\t\t"<<wt[i]<<"\t\t"<<t[i]<<"\
t\t"<<ct[i]<<endl;
       awt = awt + wt[i];
        tat= tat + t[i];
   }
   awt/=n;
        tat/=n;
        cout<<"\n\nAVG WAITING TIME=="<<awt<<endl<<"AVG
TRUN_A_TIME=="<<tat<<endl;
}
```

PRIORITY:

NON-PREEMPTIVE:

```
#include<iostream>
using namespace std;
#define M 10
int pps(int n,int p[],int bt[],int wt[],int t[],int at[],int ct[],int pp[])
{
	int x;
	float awt=0,atat=t[0];
	for(int i=1;i<n;i++)
	{
		int nop=0;
		for(int j=i;j<n;j++)
		{
			if(at[j]<=ct[i-1])
			{
				nop++;
			}
		}
		int k=i,s=i+nop;
		for(k;k<s;k++)
		{
			for(int l=k+1;l<s;l++)
			{
				if(pp[k]>pp[l])
				{
					x=bt[k];
```

```
                    bt[k]=bt[l];
                    bt[l]=x;

                    x=p[k];
                    p[k]=p[l];
                    p[l]=x;

                    x=at[k];
                    at[k]=at[l];
                    at[l]=x;

                    x=pp[k];
                    pp[k]=pp[l];
                    pp[l]=x;

                }
        }
}

ct[i]=ct[i-1]+bt[i];
t[i]=ct[i]-at[i];
wt[i]=t[i]-bt[i];
/*wt[i]=ct[i-1]-at[i];
ct[i]=wt[i]+at[i]+bt[i];
t[i]=bt[i]+wt[i]; */
awt=awt+wt[i];
atat=atat+t[i];
```

```
	}
	cout<<"\n\n\n";
	cout<<"SCHEDULING PROCESS (PRIORITY  NON-PREEMPTIVE)\n\n\n";
	cout<<"PROCESS\t\tPRIORITY\tARIVAL_TIME\tBURST_TIME\tCOMP_TIME\tTART\tWAIT_TIME\t\n";
	for(int i=0;i<n;i++)
	{
		cout<<"
"<<p[i]<<"\t\t"<<pp[i]<<"\t\t"<<at[i]<<"\t\t"<<bt[i]<<"\t\t"<<ct[i]<<"\t\t"<<t[i]<<"\t"<<wt[i]<<"\t"<<endl;
	}
	awt/=n;
	atat/=n;
	cout<<"\n\nAVG WAITING TIME=="<<awt<<endl<<"AVG TRUN_A_TIME=="<<atat<<endl;
}
int main()
{
	int x,n,p[M],bt[M],wt[M],t[M],at[M],ct[M],pp[M];
	cout<<"ENTER THE NO.OF PROCESS\n";
	cin>>n;
	for(int i=0;i<n;i++)
	{
		cout<<"ENTER THE ARIVAL_TIME , PRIORITY AND BURST_TIME FOR PROCESS "<<(i+1)<<endl;
		cout<<"ARIVAL_TIME::";
		cin>>at[i];
		cout<<"ENTER PRIORITY::";
```

```
            cin>>pp[i];
            cout<<"BURST_TIME::";
            cin>>bt[i];
            p[i]=i+1;
        }
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                if(at[i]>at[j])     //BUBBLE SORTING
                {
                    x=bt[i];
                    bt[i]=bt[j];
                    bt[j]=x;

                    x=p[i];
                    p[i]=p[j];
                    p[j]=x;

                    x=at[i];
                    at[i]=at[j];
                    at[j]=x;
                    x=pp[i];
                    pp[i]=pp[j];
                    pp[j]=x;
                }
                if(at[i]==at[j]) //if times are arival times are equal ... sorting
based on Priority of Process.
```

```
                {
                        if(pp[i]>pp[j])
                        {
                                x=bt[i];
                           bt[i]=bt[j];
                           bt[j]=x;

                           x=p[i];
                           p[i]=p[j];
                           p[j]=x;

                           x=at[i];
                           at[i]=at[j];
                           at[j]=x;

                                x=pp[i];
                                pp[i]=pp[j];
                                pp[j]=x;
                        }
                }
        }
}
//wt[0]=0;
//t[0]=bt[0];
ct[0]=at[0]+bt[0];
t[0]=ct[0]-at[0];
wt[0]=t[0]-bt[0];
```

```
        cout<<"ARIVAL_TIME\tPRIORITY\tPROCESS\tBURST_TIME\n";
        for(int i=0;i<n;i++)
        {
                cout<<" "<<at[i]<<"\t\t"<<pp[i]<<"\t\t"<<p[i]<<"\t"<<bt[i]<<endl;
        }
        pps(n,p,bt,wt,t,at,ct,pp);
}
```

PREEMPTIVE:

```
#include<iostream>
using namespace std;
#define M 10
int n,p[M],pp[M],bt[M],wt[M],t[M],at[M],ct[M],tbt[M],st[M],tm=0,i,j;
float pc=0,tat,awt;
int check(int min,int tm);
//int print(int id);
void pps()
{

        while(pc!=n)
        {
                int min=999;
                min=check(min,tm);
                bt[min]--;
      if(bt[min]==0)
      {
         pc++;
         //print(min);
         st[min]=1;
         ct[min] = tm+1;
         wt[min] = (tm+1) - at[min] - tbt[min];
         t[min] = (tm+1) - at[min];
      }
      tm++;
        }
```

```
}
int check(int min,int tm)
{
        int id=min;
      for(i=0; i<n; i++)
      {
         if(at[i]<=tm && pp[i]<min && st[i]==0 )
         {
               min=pp[i];
               id=i;
                     }
      }
      /*if(min!=id)
      print(id);*/
        return id;
}
/*int print(int id)
{
        cout<<" "<<p[id]<<"\t\t"<<bt[id]<<"\t\t"<<tm<<endl;
}*/
int main()
{

        cout<<"ENTER THE NO.OF PROCESS\n";
        cin>>n;
        for(int i=0;i<n;i++)
        {
```

```
                cout<<"ENTER THE ARIVAL_TIME AND BURST_TIME FOR PROCESS "<<(i+1)<<endl;
                cout<<"ARIVAL_TIME::";
                cin>>at[i];
                cout<<"ENTER PRIORITY::";
                cin>>pp[i];
                cout<<"BURST_TIME::";
                cin>>bt[i];
                tbt[i]=bt[i];
                p[i]=i+1;
                st[i]=0;
        }
        pps();
        cout<<"\n\n\t\t(PRIORITY SCHEDULING PREEMPTIVE)\n\n";
        cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time" <<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-time"<<endl;
    for(int i=0; i<n; i++)
    {

cout<<"p"<<i+1<<"\t\t"<<tbt[i]<<"\t\t"<<at[i]<<"\t\t"<<wt[i]<<"\t\t"<<t[i]<<"\t\t"<<ct[i]<<endl;
        awt = awt + wt[i];
         tat= tat + t[i];
    }
    awt/=n;
         tat/=n;
        cout<<"\n\nAVG WAITING TIME=="<<awt<<endl<<"AVG TRUN_A_TIME=="<<tat<<endl;
}
```

ROUND-ROBIN:

```
#include<iostream>
using namespace std;
# define M 10
int main()
{
	int x,n,i,p[M],bt[M],wt[M],t[M],at[M],ct[M],tm=0,ti,pc=0,st[M],tbt[M];
	float awt=0,tat=0;
	cout<<"ENTER THE NO.OF PROCESS\n";
	cin>>n;
	for(int i=0;i<n;i++)
	{
		cout<<"ENTER THE ARIVAL_TIME AND BURST_TIME FOR PROCESS "<<(i+1)<<endl;
		cout<<"ARIVAL_TIME::";
		cin>>at[i];
		cout<<"BURST_TIME::";
		cin>>bt[i];
		tbt[i]=bt[i];
		p[i]=i+1;
		st[i]=0;
	}
	for(int i=0;i<n;i++)
	{
		for(int j=i+1;j<n;j++)
		{
			if(at[i]>at[j])	//BUBBLE SORTING
			{
```

```
                x=bt[i];
                bt[i]=bt[j];
                bt[j]=x;

                x=tbt[i];
                tbt[i]=tbt[j];
                tbt[j]=x;

                x=p[i];
                p[i]=p[j];
                p[j]=x;

                x=at[i];
                at[i]=at[j];
                at[j]=x;
            }

        }
}
cout<<"\n\nENTER THE TIME SLICE FOR EACH PROCESS\n\t";
cin>>ti;
while(pc!=n)
for(i=0;(i<n);i++)
{
        do
        {
            if(bt[i]>0 && st[i]!=1)
```

```
                {
                        bt[i]--;
                   if(bt[i]==0)
                   {
                           pc++;
                           st[i]=1;
                           ct[i] = tm+1;
            wt[i] = (tm+1) - at[i] - tbt[i];
            t[i] = (tm+1) - at[i];
                        }
                   tm++;
                }
                else
                break;
             }while((tm%ti!=0)&&(st[i]!=1));
        }
        cout<<"\t\t\tROUND ROBIN-SCHEDULING\n\n";
        cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time"
<<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-
time"<<endl;
   for(int i=0; i<n; i++)
   {

cout<<"P"<<i+1<<"\t\t"<<tbt[i]<<"\t\t"<<at[i]<<"\t\t"<<wt[i]<<"\t\t"<<t[i]<<"\
t\t"<<ct[i]<<endl;
       awt = awt + wt[i];
       tat= tat + t[i];
   }
   awt/=n;
```

```
        tat/=n;

        cout<<"\n\nAVG WAITING TIME=="<<awt<<endl<<"AVG
TRUN_A_TIME=="<<tat<<endl;


}
```