Non Preemptive Priority:

```
#include <stdlib.h>
#include <stdio.h>
void main()
{
int pn = 0;
int CPU = 0;
int allTime = 0;
printf("No. of Processes: ");
scanf("%d",&pn);
int AT[pn];
int ATt[pn];
int NoP = pn;
int PT[pn];
int PP[pn];
int PPt[pn];
int waittingTime[pn];
int turnaroundTime[pn];
int i=0;
for(i=0 ;i<pn ;i++){
printf("\nBurst Time P%d: ",i+1);
scanf("%d",&PT[i]);
printf("Piriorty P%d: ",i+1);
scanf("%d",&PP[i]);
PPt[i] = PP[i];
printf("Arrival Time P%d: ",i+1);
scanf("%d",&AT[i]);
ATt[i] = AT[i];
}
int LAT = 0;
for(i = 0; i < pn; i++)
if(AT[i] > LAT)
LAT = AT[i];
int MAX_P = 0;
for(i = 0; i < pn; i++)
if(PPt[i] > MAX_P)
MAX_P = PPt[i];
int ATi = 0;
int P1 = PPt[0];
int P2 = PPt[0];
int j = -1;
while(NoP > 0 && CPU <= 1000){
for(i = 0; i < pn; i++){
if((ATt[i] <= CPU) && (ATt[i] != (LAT+10))){
if(PPt[i] != (MAX_P+1)){
P2 = PPt[i];
j= 1;
if(P2 < P1){
j= 1;
ATi = i;
P1 = PPt[i];
P2 = PPt[i];
}
}
}
}
if(j == -1){
CPU = CPU+1;
```

```
continue;
}else{
waittingTime[ATi] = CPU - ATt[ATi];
 CPU = CPU + PT[ATi];
turnaroundTime[ATi] = CPU - ATt[ATi];
ATt[ATi] = LAT +10;
j = -1;
PPt[ATi] = MAX_P + 1;
ATi = 0;
P1 = MAX_P+1;
P2 = MAX_P+1;
NoP = NoP - 1;
}
}
printf("\nPN\tPT\tPP\tAT\tWT\tTT\n\n");
for(i = 0; i < pn; i++){
printf("P%d\t%d\t%d\t%d\t%d\t%d\n",i+1,PT[i],PP[i],AT[i],waittingTime[i],
turnaroundTime[i]);
}
int AvgWT = 0;
int AVGTaT = 0;
for(i = 0; i < pn; i++){
AvgWT = waittingTime[i] + AvgWT;
AVGTaT = turnaroundTime[i] + AVGTaT;
}
printf("Avg Waitting Time = %d\nAvg Turnaround Time =
%d\n",AvgWT/pn,AVGTaT/pn);
return 0;
}
```

Preemptive Priority:

```
#include<stdio.h>
struct process
{
int WT,AT,BT,TAT,PT;
};
struct process a[10];
int main()
{
int n,temp[10],t,count=0,short_p;
float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
printf("No. of Processes: \n");
scanf("%d",&n);
printf("AT BT PT\n");
int i;
for(i=0;i<n;i++)
{
scanf("%d %d %d",&a[i].AT,&a[i].BT,&a[i].PT);
temp[i]=a[i].BT;
}
a[9].PT=10000;
for(t=0;count!=n;t++)
{
short_p=9;
for(i=0;i<n;i++)
```

```
{
if(a[short_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)
{
short_p=i;
}
}
a[short_p].BT=a[short_p].BT-1;
if(a[short_p].BT==0)
{
count++;
a[short_p].WT=t+1-a[short_p].AT-temp[short_p];
a[short_p].TAT=t+1-a[short_p].AT;
total_WT=total_WT+a[short_p].WT;
total_TAT=total_TAT+a[short_p].TAT;
}
}
Avg_WT=total_WT/n;
Avg_TAT=total_TAT/n;
printf("ID WT TAT\n");
for(i=0;i<n;i++)
{
printf("%d %d\t %d\n",i+1,a[i].WT,a[i].TAT);
}
printf("Avg Waiting Time: %f\n",Avg_WT);
printf("Avg Turnaround Time: %f\n",Avg_TAT);
return 0;
}
```

Shortest Job First:

```
#include<stdio.h>
int main()
{
int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
float avg_wt,avg_tat;
printf("Enter number of process:");
scanf("%d",&n);
printf("\nEnter Burst Time:\n");
for(i=0;i<n;i++)
{
printf("P%d:",i+1);
scanf("%d",&bt[i]);
p[i]=i+1;
}
for(i=0;i<n;i++)
{
pos=i;
for(j=i+1;j<n;j++)
{
if(bt[j]<bt[pos])
pos=j;
}
temp=bt[i];
bt[i]=bt[pos];
bt[pos]=temp;
temp=p[i];
```

```
p[i]=p[pos];
p[pos]=temp;
}
wt[0]=0;
for(i=1;i<n;i++)
{
wt[i]=0;
for(j=0;j<i;j++)
wt[i]+=bt[j];
total+=wt[i];
}
avg_wt=(float)total/n;
total=0;
printf("\n Process \t    Burst Time    \t Waiting Time \t Turnaround
Time");
for(i=0;i<n;i++)
{
tat[i]=bt[i]+wt[i];
total+=tat[i];
printf("\n P%d \t\t  %d \t\t    %d \t\t \t%d",p[i],bt[i],wt[i],tat[i]);
}
avg_tat=(float)total/n;
printf("\n\nAverage Waiting Time=%f",avg_wt);
printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```

SRTF:

```
#include <stdio.h>
int main()
{
int a[10],b[10],x[10],i,j,smallest,count=0,time,n;
double avg=0,tt=0,end;
printf("No. of Processes:\n");
scanf("%d",&n);
printf("Arrival Times: \n");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("Burst Times: \n");
for(i=0;i<n;i++)
scanf("%d",&b[i]);
for(i=0;i<n;i++)
x[i]=b[i];
b[9]=9999;
for(time=0;count!=n;time++)
{
smallest=9;
for(i=0;i<n;i++)
{
if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
smallest=i;
}
b[smallest]--;
if(b[smallest]==0)
{
count++;
end=time+1;
```

```
avg=avg+end-a[smallest]-x[smallest];
tt= tt+end-a[smallest];
}
}
printf("\n\nAverage waiting time = %lf\n",avg/n);
printf("Average Turnaround time = %lf",tt/n);
return 0;
}
```

Round Robin:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>


typedef struct
{
        int pid;
        int arrival_time;
        int burst_time;
        int rem_time;
        int priority;
}Process;

Process* Entry()
{
printf("No. of Processes:");
int n;
scanf("%d", &n);
Process* p = (Process*) malloc( n+1 * sizeof(Process) );
printf("Enter PID, Arrival Time, Burst Time, Priority\n");
for(int i=0;i<n;i++){
printf("Process %d :",i+1);
scanf("%d %d %d %d", &p[i].pid, &p[i].arrival_time, &p[i].burst_time,
&p[i].priority);
p[i].rem_time=p[i].burst_time;
}
p[n].pid=-1;
return p;
}

void RR()
{
Process* p = Entry();
int n=0;
while(p[n].pid!=-1)
n++;
int q;
printf("Enter Time Slice:");
scanf("%d", &q);
int timer=0;
int done=0;
float TAT=0;
float WAT=0;
```

```
while(done!=n)
{
for(int i=0;i<n;i++){
        int count=q;
        int start=timer;
while(p[i].rem_time>0 && p[i].arrival_time<=timer && count>0){
        timer++;
 p[i].rem_time--;
 count--;
}
if(start!=timer){
printf("%d -> (%d %d)\n", p[i].pid, start, timer);
if(p[i].rem_time==0){
TAT+=(timer-p[i].arrival_time);
WAT+=(timer-p[i].arrival_time-p[i].burst_time);;
done++;
}
}
}
}
TAT=TAT/n;
WAT=WAT/n;
printf("Turnaround Time = %0.2f ms \n", TAT);
printf("Waiting Time = %0.2f ms\n\n\n", WAT);
}
int main(){
        RR();
        return 0;
}
```

FCFS :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void swap(int *xp, int *yp)
{
            int temp = *xp;
                *xp = *yp;
                    *yp = temp;
}
void bubbleSort(int arr[], int time[],int priority[],int n)
{
           int i, j;
              for (i = 0; i < n-1; i++){
for (j = 0; j < n-i-1; j++){
  if (arr[j] > arr[j+1]){
  swap(&arr[j], &arr[j+1]);
   swap(&time[j],&time[j+1]);
    swap(&priority[j],&priority[j+1]);
}     }
            }
}
void fcfs(int arr[],int time[],int priority[],int n)
{
         bubbleSort(arr,time,priority,n);
```

```
          int wait[n];
           int tottime=0;
            int turn[n];
             memset( wait, 0, n*sizeof(int) );
              memset( turn,0, n*sizeof(int) );
               for (int i = 0;i<n;i++)
               {
printf("Process with priority %d is executing\n",priority[i]);
 if (i==0)
{
wait[i]=arr[i];
tottime+=time[i];
turn[i]=time[i]-arr[i];
}
else
{
wait[i]=tottime-arr[i];
tottime+=time[i];
turn[i]=tottime-arr[i];
 }
printf("%d  %d  %d\n",wait[i],tottime,turn[i]);
}
float avgwait=0.0,avgburst=0.0;
 for (int i=0;i<n;i++)
 {
avgburst+=turn[i];
avgwait+=wait[i];
 }
avgwait/=n;avgburst/=n;
printf("Average Burst Time %f ",avgburst);
printf("\nAverage Wait Time %f \n",avgwait );
}
int main()
{
int arr[100],time[100],priority[100],n;
printf("Enter the number of processes\n");
scanf("%d",&n);
for (int i =0 ;i < n;i++)
{

printf("Arrival Time time for process P%d ",i);
scanf("%d",&arr[i]);

printf("Burst time for process P%d ",i);
scanf("%d",&time[i]);

printf("Priority for process P%d ",i);
scanf("%d",&priority[i]);
}
fcfs(arr,time,priority,n);
}
```