Saurabh Kumar Singh(16100051)

SJF Scheduling

ALGORITHM::

Mode:- Non-Preemptive

Criteria:- Burst Time

1- Sort all the processes in increasing order according to burst time.

2- Then FCFS

CODE::

```
#include<stdio.h>
int main()
{
   int at[100],bt[100],ct[100],tat[100],wt[100],n,i,j,k,pid[100],l[100];
    printf("\nEnter the number of process : ");
    scanf("%d",&n);

    printf("Enter the Arrival time and Burst time.\n\n");
    printf("\tArrival_Time Burst_Time\n");
   for(i=0;i<n;i++){
      scanf("%d",&at[i]);
        scanf("%d",&bt[i]);
      ct[i]=0;
      tat[i]=0;
      wt[i]=0;
      pid[i]=i;
      l[i]=at[i];
   }
   ct[-1]=0;
for(i=0;i<n;i++)
        {
            for(j=i+1;j<n;j++)
            {
               if(at[i]>at[j])
                {
                  int t=at[i];
                  at[i]=at[j];
                  at[j]=t;
                  t=bt[i];
                  bt[i]=bt[j];
                  bt[j]=t;
                  t=pid[i];
                  pid[i]=pid[j];pid[j]=t;
                }
            }
        }

       for(i=0;i<n;i++)
        {
           for(j=i+1;j<n;j++)
            {
               if(bt[i]>bt[j]&&at[i]==at[j])
                {
                   int t=bt[i];
                   bt[i]=bt[j];
                   bt[j]=t;
                   t=pid[i];
                   pid[i]=pid[j];pid[j]=t;

                }
            }
        }


   int kl=0;
   int y=0;
printf("\npid\t\tarrival time\tburst Time\tcompletion time\tTurnaround Time\tWaiting
Time");
   //completion time
   for(k=0;k<n;k++)
   {


      int r=0;
      if((ct[k-1]+kl)>=at[k])
      {
         ct[k]=ct[k-1]+bt[k]+kl;
         kl=0;
         y=ct[k];
         r=1;
      }
      else
      {
         kl++;
         k=k-1;
         r=0;
      }
      if(r==1){
```

```
      for(i=k+1;i<n;i++)
        {
           for(j=i+1;j<n;j++)
            {
               if(bt[i]>bt[j]&&at[j]<=y&&at[j]<=y)
                {
                   int t=bt[i];
                   bt[i]=bt[j];
                   bt[j]=t;
                   t=pid[i];
                   pid[i]=pid[j];pid[j]=t;
                   t=at[i];
                  at[i]=at[j];
                  at[j]=t;

                }
            }
        }
}

   }

   //turn around time
   for(i=0;i<n;i++)
   {
      tat[i]=ct[i]-at[i];
   }
    for(i=0;i<n;i++)
   {
      wt[i]=tat[i]-bt[i];
   }
   for(i=0;i<n;i++)
   {

printf("\nP[%d]\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d",i+1,at[i],bt[i],ct[i],tat[i],wt[i]);
      printf("\n");
       }

 float avgtat=0,avgwt=0;
  for(i=0;i<n;i++)
  {
   avgtat=avgtat+tat[i];
   avgwt=avgwt+wt[i];
  }
      printf("\navg tat is : %f",avgtat/n);
      printf("\navg wt is: %f",avgwt/n);
   return 0;
```

```
}
```

INPUT::

Enter the number of process : 6
Enter the Arrival time and Burst time.

```
     Arrival_Time Burst_Time
1 3    4 4    1 2    2 4    7 2    8 1
```

OUTPUT::

```
pid          arrival time    burst Time      completion time Turnaround Time Waiting Time
P[1]        1            2            3            2            0

P[2]        1            3            6            5            2

P[3]        2            4            10           8            4

P[4]        8            1            11           3            2

P[5]        7            2            13           6            4

P[6]        4            4            17           13           9

avg tat is : 6.166667
avg wt is: 3.500000
```