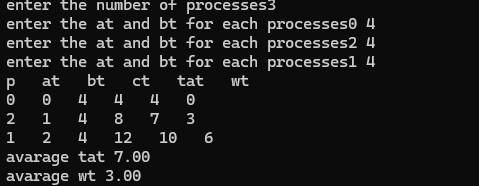
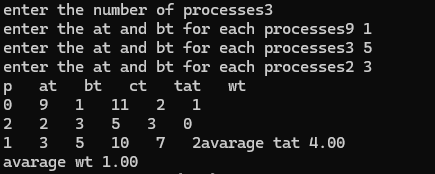
FCFS

#include<stdio.h>   
#include <stdlib.h>  
void sorting(int at[],int bt[],int p[],int n)  
{ int temp;  
   for(int i=0;i<n;i++)  
   {  
       for(int j=i+1;j<n;j++)  
       {  
           if(at[i]>at[j])  
           {  
             temp=at[i];  
             at[i]=at[j];  
             at[j]=temp;  
  
             temp=bt[i];  
             bt[i]=bt[j];  
             bt[j]=temp;  
  
             temp=p[i];  
             p[i]=p[j];  
             p[j]=temp;  
           }  
       }  
   }  
}  
void ctandst(int at[],int bt[],int ct[],int st[],int n)  
{   ct[0]=at[0]+bt[0];  
    st[0]=at[0];  
    for(int i=1;i<n;i++)  
    {  
        if(ct[i-1]<at[i])  
        {st[i]=at[i];  
            ct[i]=at[i]+bt[i];  
  
        }  
  
        else  
          {  
              st[i]=ct[i-1];  
              ct[i]=st[i]+bt[i];  
          }  
    }  
}  
void turnat(int ct[],int at[],int tat[],int n)  
{  
    for(int i=0;i<n;i++)  
    {  
        tat[i]=ct[i]-at[i];  
    }  
}  
void waitt(int tat[],int bt[],int wt[],int n)  
{  
    for(int i=0;i<n;i++)  
    {  
        wt[i]=tat[i]-bt[i];  
    }  
}  
  
int main()  
{  
  
    int n;  
    printf("enter the number of processes");  
    scanf("%d",&n);  
    int p[n],at[n],bt[n],ct[n],rt[n],tat[n],wt[n],st[n];  
    for(int i=0;i<n;i++)  
    {  
      p[i]=i;  
      printf("enter the at and bt for each processes");  
      scanf("%d %d",&at[i],&bt[i]);  
  
    }  
    sorting(at,bt,p,n);  
    ctandst(at,bt,ct,st,n);  
    turnat(ct,at,tat,n);  
    waitt(tat,bt,wt,n);  
  
    int sum1=0,sum2=0;  
    for(int i=0;i<n;i++)  
    {  
        sum1+=tat[i];  
        sum2+=wt[i];  
  
    }  
    printf("p   at   bt   ct   tat   wt");  
    for(int i=0;i<n;i++)  
    {  
        printf("\n%d   %d   %d   %d   %d   %d",p[i],at[i],bt[i],ct[i],tat[i],wt[i]);  
    }  
    printf("\navarage tat %.2f",(float)sum1/n);  
     printf("\navarage wt %.2f",(float)sum2/n);  
   return 0;  
}



SJF-preemtive

#include<stdio.h>  
#include <stdlib.h>  
void sorting(int at[],int bt[],int p[],int n)  
{ int temp;  
   for(int i=0;i<n;i++)  
   {  
       for(int j=i+1;j<n;j++)  
       {  
           if(bt[i]>bt[j])  
           {  
             temp=at[i];  
             at[i]=at[j];  
             at[j]=temp;  
  
             temp=bt[i];  
             bt[i]=bt[j];  
             bt[j]=temp;  
  
             temp=p[i];  
             p[i]=p[j];  
             p[j]=temp;  
           }  
       }  
   }  
}  
  
void ctandst(int at[], int bt[], int ct[], int st[], int n) {  
    int remaining = n;  
    int completed = 0;  
    int min\_bt;  
    int current\_time = 0;  
    int idx;  
  
    // Initially set start time and completion time for each process to -1 (not yet computed)  
    for (int i = 0; i < n; i++) {  
        st[i] = -1;  
        ct[i] = -1;  
    }  
  
    // Keep executing processes based on shortest burst time  
    while (completed < n) {  
        min\_bt = 9999; // Assume the burst time is always smaller than this value  
        idx = -1;  
  
        // Find the process with the smallest burst time that is ready to execute  
        for (int i = 0; i < n; i++) {  
            if (st[i] == -1 && at[i] <= current\_time && bt[i] < min\_bt) {  
                min\_bt = bt[i];  
                idx = i;  
            }  
        }  
  
        // If no process is found (i.e., all remaining processes are in the future), skip to the next arrival time  
        if (idx == -1) {  
            current\_time++;  
            continue;  
        }  
  
        // Set the start time and completion time for the selected process  
        st[idx] = current\_time;  
        ct[idx] = current\_time + bt[idx];  
        current\_time = ct[idx]; // Update current time to the completion time of the executed process  
  
        // Mark the process as completed  
        completed++;  
    }  
}  
void turnat(int ct[],int at[],int tat[],int n)  
{  
    for(int i=0;i<n;i++)  
    {  
        tat[i]=ct[i]-at[i];  
    }  
}  
void waitt(int tat[],int bt[],int wt[],int n)  
{  
    for(int i=0;i<n;i++)  
    {  
        wt[i]=tat[i]-bt[i];  
    }  
}  
  
int main()  
{  
  
    int n;  
    printf("enter the number of processes");  
    scanf("%d",&n);  
    int p[n],at[n],bt[n],ct[n],rt[n],tat[n],wt[n],st[n];  
    for(int i=0;i<n;i++)  
    {  
      p[i]=i;  
      printf("enter the at and bt for each processes");  
      scanf("%d %d",&at[i],&bt[i]);  
  
    }  
    sorting(at,bt,p,n);  
    ctandst(at,bt,ct,st,n);  
    turnat(ct,at,tat,n);  
    waitt(tat,bt,wt,n);  
     printf("p   at   bt   ct   tat   wt");  
    for(int i=0;i<n;i++)  
    {  
        printf("\n%d   %d   %d   %d   %d   %d",p[i],at[i],bt[i],ct[i],tat[i],wt[i]);  
    }  
  
    int sum1=0,sum2=0;  
    for(int i=0;i<n;i++)  
    {  
        sum1+=tat[i];  
        sum2+=wt[i];  
  
    }  
    printf("avarage tat %.2f",(float)sum1/n);  
     printf("\navarage wt %.2f",(float)sum2/n);  
   return 0;  
}



SJF-non preemtive

#include <stdio.h>

#include <stdlib.h>

void sorting(int at[], int bt[], int p[], int n) {

int temp;

for (int i = 0; i < n; i++) {

for (int j = i + 1; j < n; j++) {

if (bt[i] > bt[j]) {

temp = at[i];

at[i] = at[j];

at[j] = temp;

temp = bt[i];

bt[i] = bt[j];

bt[j] = temp;

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}

void ctandst(int at[], int bt[], int ct[], int st[], int n) {

int remaining = n;

int completed = 0;

int min\_bt;

int current\_time = 0;

int idx;

for (int i = 0; i < n; i++) {

st[i] = -1;

ct[i] = -1;

}

// Execute processes in order of shortest burst time, considering the arrival time

while (completed < n) {

min\_bt = 9999; // Assume the burst time is always smaller than this value

idx = -1;

// Find the process with the smallest burst time that is ready to execute

for (int i = 0; i < n; i++) {

if (st[i] == -1 && at[i] <= current\_time && bt[i] < min\_bt) {

min\_bt = bt[i];

idx = i;

}

}

// If no process is found (i.e., all remaining processes are in the future), skip to the next arrival time

if (idx == -1) {

current\_time++;

continue;

}

// Set the start time and completion time for the selected process

st[idx] = current\_time;

ct[idx] = current\_time + bt[idx];

current\_time = ct[idx]; // Update current time to the completion time of the executed process

// Mark the process as completed

completed++;

}

}

void turnat(int ct[], int at[], int tat[], int n) {

for (int i = 0; i < n; i++) {

tat[i] = ct[i] - at[i];

}

}

void waitt(int tat[], int bt[], int wt[], int n) {

for (int i = 0; i < n; i++) {

wt[i] = tat[i] - bt[i];

}

}

int main() {

int n;

printf("Enter the number of processes: ");

scanf("%d", &n);

int p[n], at[n], bt[n], ct[n], rt[n], tat[n], wt[n], st[n];

// Input the arrival and burst times for each process

for (int i = 0; i < n; i++) {

p[i] = i;

printf("Enter the arrival time and burst time for process %d: ", i);

scanf("%d %d", &at[i], &bt[i]);

}

sorting(at, bt, p, n);

ctandst(at, bt, ct, st, n);

turnat(ct, at, tat, n);

waitt(tat, bt, wt, n);

printf("\np at bt ct tat wt\n");

for (int i = 0; i < n; i++) {

printf("%d %d %d %d %d %d\n", p[i], at[i], bt[i], ct[i], tat[i], wt[i]);

}

int sum1 = 0, sum2 = 0;

for (int i = 0; i < n; i++) {

sum1 += tat[i];

sum2 += wt[i];

}

printf("\nAverage Turnaround Time: %.2f", (float)sum1 / n);

printf("\nAverage Waiting Time: %.2f", (float)sum2 / n);

return 0;

}

