**1)**

**Aim:** Write a Program to demonstrate First Come First Serve CPU scheduling

**Program:**

#include<stdio.h>

int main()

{

int p[10],at[10],bt[10],ct[10],tat[10],wt[10],i,j,temp=0,n;

float awt=0,atat=0;

printf("enter no of proccess you want:");

scanf("%d",&n);

printf("enter %d process:",n);

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

{

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

}

printf("enter %d arrival time:",n);

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

{

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

}

printf("enter %d burst time:",n);

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

{

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

}

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

{

for(j=0;j<(n-i);j++)

{

if(at[j]>at[j+1])

{

temp=p[j+1];

p[j+1]=p[j];

p[j]=temp;

temp=at[j+1];

at[j+1]=at[j];

at[j]=temp;

temp=bt[j+1];

bt[j+1]=bt[j];

bt[j]=temp;

}

}

}

ct[0]=at[0]+bt[0];

for(i=1;i<n;i++)

{

temp=0;

if(ct[i-1]<at[i])

{

temp=at[i]-ct[i-1];

}

ct[i]=ct[i-1]+bt[i]+temp;

}

printf("\np\t A.T\t B.T\t C.T\t TAT\t WT");

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

{

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

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

atat+=tat[i];

awt+=wt[i];

}

atat=atat/n;

awt=awt/n;

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

{

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

}

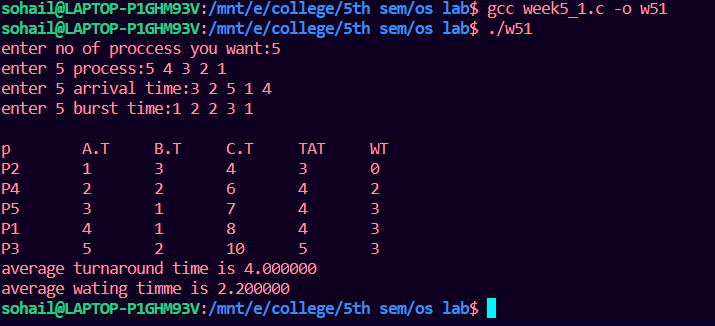
printf("\naverage turnaround time is %f",atat);

printf("\naverage wating timme is %f\n",awt);

return 0;

}

**Output:**

****

**2)**

**Aim:** Write a Program to demonstrate Shortest Job First or Shortest Job next CPU scheduling

**Program:(preemptive)**

#include <stdio.h>

int main()

{

int arrival\_time[10], burst\_time[10], temp[10];

int i, smallest, count = 0, time, limit;

double wait\_time = 0, turnaround\_time = 0, end;

float average\_waiting\_time, average\_turnaround\_time;

printf("\nEnter the Total Number of Processes:\t");

scanf("%d", &limit);

printf("\nEnter Details of %d Processes\n", limit);

for(i = 0; i < limit; i++)

{

printf("\nEnter Arrival Time:\t");

scanf("%d", &arrival\_time[i]);

printf("Enter Burst Time:\t");

scanf("%d", &burst\_time[i]);

temp[i] = burst\_time[i];

}

burst\_time[9] = 9999;

for(time = 0; count != limit; time++)

{

smallest = 9;

for(i = 0; i < limit; i++)

{

if(arrival\_time[i] <= time && burst\_time[i] < burst\_time[smallest] && burst\_time[i] > 0)

{

smallest = i;

}

}

burst\_time[smallest]--;

if(burst\_time[smallest] == 0)

{

count++;

end = time + 1;

wait\_time = wait\_time + end - arrival\_time[smallest] - temp[smallest];

turnaround\_time = turnaround\_time + end - arrival\_time[smallest];

}

}

average\_waiting\_time = wait\_time / limit;

average\_turnaround\_time = turnaround\_time / limit;

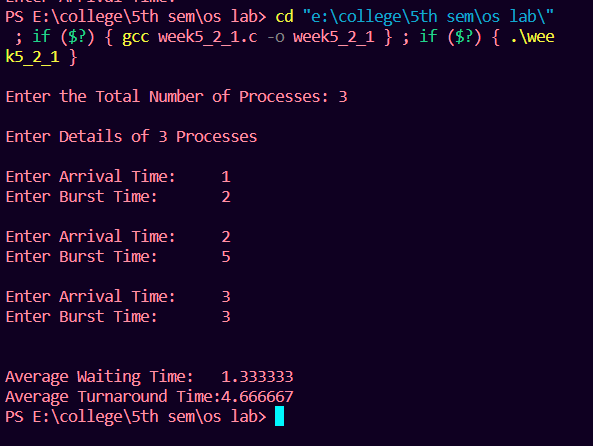
printf("\n\nAverage Waiting Time:\t%lf\n", average\_waiting\_time);

printf("Average Turnaround Time:%lf\n", average\_turnaround\_time);

return 0;

}

**Output:**



**Program(non preemptive):**

#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 processes:");

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;

}

//sorting of burst times

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("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");

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

{

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

total+=tat[i];

printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

}

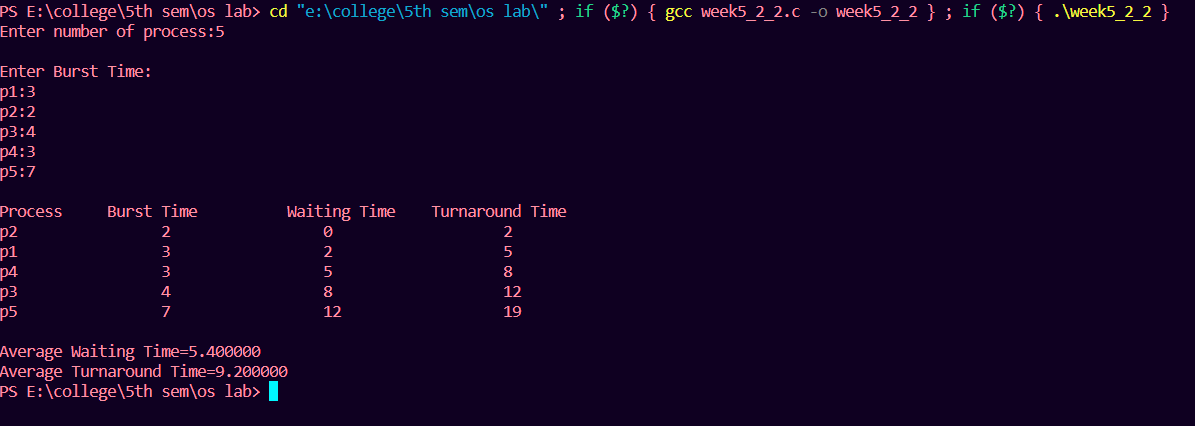
avg\_tat=(float)total/n;

printf("\n\nAverage Waiting Time=%f",avg\_wt);

printf("\nAverage Turnaround Time=%f\n",avg\_tat);

}

**O****utput:**

****

**3)**

**Aim:** Write a Program to demonstrate Round Robin CPU scheduling

**Program:**

#include<stdio.h>

int main()

{

int i, limit, total = 0, x, counter = 0, time\_quantum;

int wait\_time = 0, turnaround\_time = 0, arrival\_time[10], burst\_time[10], temp[10];

float average\_wait\_time, average\_turnaround\_time;

printf("\nEnter Total Number of Processes:\t");

scanf("%d", &limit);

x = limit;

for(i = 0; i < limit; i++)

{

printf("\nEnter Details of Process[%d]\n", i + 1);

printf("Arrival Time:\t");

scanf("%d", &arrival\_time[i]);

printf("Burst Time:\t");

scanf("%d", &burst\_time[i]);

temp[i] = burst\_time[i];

}

printf("\nEnter Time Quantum:\t");

scanf("%d", &time\_quantum);

printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting Time\n");

for(total = 0, i = 0; x != 0;)

{

if(temp[i] <= time\_quantum && temp[i] > 0)

{

total = total + temp[i];

temp[i] = 0;

counter = 1;

}

else if(temp[i] > 0)

{

temp[i] = temp[i] - time\_quantum;

total = total + time\_quantum;

}

if(temp[i] == 0 && counter == 1)

{

x--;

printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1, burst\_time[i], total - arrival\_time[i], total - arrival\_time[i] - burst\_time[i]);

wait\_time = wait\_time + total - arrival\_time[i] - burst\_time[i];

turnaround\_time = turnaround\_time + total - arrival\_time[i];

counter = 0;

}

if(i == limit - 1)

{

i = 0;

}

else if(arrival\_time[i + 1] <= total)

{

i++;

}

else

{

i = 0;

}

}

average\_wait\_time = wait\_time \* 1.0 / limit;

average\_turnaround\_time = turnaround\_time \* 1.0 / limit;

printf("\n\nAverage Waiting Time:\t%f", average\_wait\_time);

printf("\nAvg Turnaround Time:\t%f\n", average\_turnaround\_time);

return 0;

}

**Output:**

****

**4)**

**Aim:** Write a Program to demonstrate Priority CPU Scheduling

**Program(preemptive):**

#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("Enter the number of the process\n");

scanf("%d",&n);

printf("Enter the arrival time , burst time and priority of the process\n");

printf("AT BT PT\n");

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

{

scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

// copying the burst time in

// a temp array fot futher use

temp[i]=a[i].BT;

}

// we initialize the burst time

// of a process with maximum

a[9].PT=10000;

for(t=0;count!=n;t++)

{

short\_p=9;

for(int 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 any process is completed

if(a[short\_p].BT==0)

{

// one process is completed

// so count increases by 1

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 calculation

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;

// printing of the answer

printf("ID WT TAT\n");

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

{

printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);

}

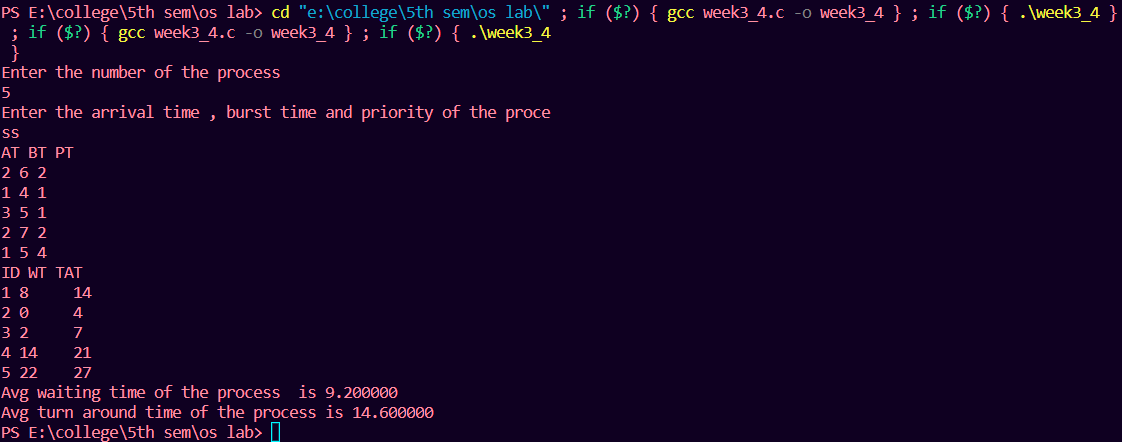
printf("Avg waiting time of the process is %f\n",Avg\_WT);

printf("Avg turn around time of the process is %f\n",Avg\_TAT);

return 0;

}

**Output:**

****

**Program(Non preemptive):**

#include<stdio.h>

struct process

{

int id,WT,AT,BT,TAT,PR;

};

struct process a[10];

// function for swapping

void swap(int \*b,int \*c)

{

int tem;

tem=\*c;

\*c=\*b;

\*b=tem;

}

//Driver function

int main()

{

int n,check\_ar=0;

int Cmp\_time=0;

float Total\_WT=0,Total\_TAT=0,Avg\_WT,Avg\_TAT;

printf("Enter the number of process \n");

scanf("%d",&n);

printf("Enter the Arrival time , Burst time and priority of the process\n");

printf("AT BT PR\n");

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

{

scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PR);

a[i].id=i+1;

// here we are checking that arrival time

// of the process are same or different

if(i==0)

check\_ar=a[i].AT;

if(check\_ar!=a[i].AT )

check\_ar=1;

}

// if process are arrived at the different time

// then sort the process on the basis of AT

if(check\_ar!=0)

{

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

{

for(int j=0;j<n-i-1;j++)

{

if(a[j].AT>a[j+1].AT)

{

swap(&a[j].id,&a[j+1].id);

swap(&a[j].AT,&a[j+1].AT);

swap(&a[j].BT,&a[j+1].BT);

swap(&a[j].PR,&a[j+1].PR);

}}}

// logic of Priority scheduling ( non preemptive) algo

// if all the process are arrived at different time

if(check\_ar!=0)

{

a[0].WT=a[0].AT;

a[0].TAT=a[0].BT-a[0].AT;

// cmp\_time for completion time

Cmp\_time=a[0].TAT;

Total\_WT=Total\_WT+a[0].WT;

Total\_TAT=Total\_TAT+a[0].TAT;

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

{

int min=a[i].PR;

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

{

if(min>a[j].PR && a[j].AT<=Cmp\_time)

{

min=a[j].PR;

swap(&a[i].id,&a[j].id);

swap(&a[i].AT,&a[j].AT);

swap(&a[i].BT,&a[j].BT);

swap(&a[i].PR,&a[j].PR);

}

}

a[i].WT=Cmp\_time-a[i].AT;

Total\_WT=Total\_WT+a[i].WT;

// completion time of the process

Cmp\_time=Cmp\_time+a[i].BT;

// Turn Around Time of the process

// compl-Arival

a[i].TAT=Cmp\_time-a[i].AT;

Total\_TAT=Total\_TAT+a[i].TAT;

}

}

// if all the process are arrived at same time

else

{

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

{

int min=a[i].PR;

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

{

if(min>a[j].PR && a[j].AT<=Cmp\_time)

{

min=a[j].PR;

swap(&a[i].id,&a[j].id);

swap(&a[i].AT,&a[j].AT);

swap(&a[i].BT,&a[j].BT);

swap(&a[i].PR,&a[j].PR);

}

}

a[i].WT=Cmp\_time-a[i].AT;

// completion time of the process

Cmp\_time=Cmp\_time+a[i].BT;

// Turn Around Time of the process

// compl-Arrival

a[i].TAT=Cmp\_time-a[i].AT;

Total\_WT=Total\_WT+a[i].WT;

Total\_TAT=Total\_TAT+a[i].TAT;

}

}

Avg\_WT=Total\_WT/n;

Avg\_TAT=Total\_TAT/n;

// Printing of the results

printf("The process are\n");

printf("ID WT TAT\n");

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

{

printf("%d\t%d\t%d\n",a[i].id,a[i].WT,a[i].TAT);

}

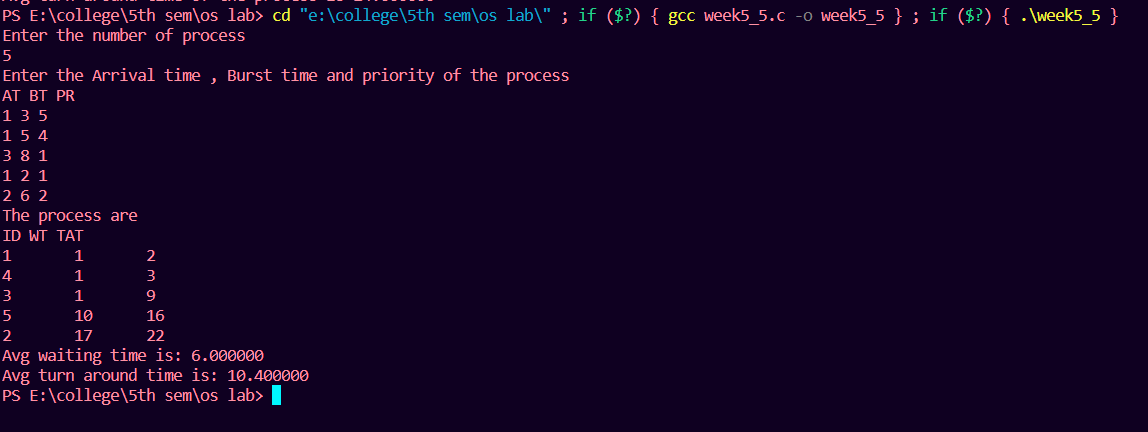
printf("Avg waiting time is: %f\n",Avg\_WT);

printf("Avg turn around time is: %f",Avg\_TAT);

return 0;

}

**Output:**

****