1]

//Programmer Name: Sharvil Prabhudesai 20co41

//Program Title : To implement First come first serve without arrival time.

#include<stdio.h>

int main()

{

int i,j,n ;

printf("FIRST COME FIRST SERVE WITHOUT ARRIVAL\n\n");

printf("Enter Number of Processes : ");

scanf("%d",&n);

int p[n];

int bt[n];

int wt[n];

int tt[n];

float avg\_tt=0;

float avg\_wt=0;

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

p[i]=i+1;

printf("Enter the Burst time of Process P%d : ",p[i]);

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

}

printf("\n\n");

wt[0] = 0; //need to be initialised because of formula

tt[0]=bt[0];

avg\_wt+=wt[0];

avg\_tt+=tt[0];

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

wt[i]=wt[i-1] + bt[i-1];

tt[i]=tt[i-1]+bt[i];

avg\_wt+=wt[i];

avg\_tt+=tt[i];

}

printf("\n\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("| Process | Burst Time | Waiting Time | Turnaround time |\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

printf("| %d | %3d | %3d | %3d | \n",p[i],bt[i],wt[i],tt[i]);

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

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

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

printf("\n\nGANTT CHART ");

printf("\n\n");

// print top bar

printf(" ");

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

for(j=0; j<bt[i]; j++){

printf("--");

}

printf(" ");

}

printf("\n|");

// printing process id in the middle

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

for(j=0; j<bt[i]- 1; j++){

printf(" ");

}

printf(" %d",p[i]);

for(j=0; j<bt[i] - 1; j++) {

printf(" ");

}

printf("|");

}

printf("\n ");

// printing bottom bar

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

for(j=0; j<bt[i]; j++) {

printf("--");

}

printf(" ");

}

printf("\n");

// printing the time line

printf("0");

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

for(j=0; j<bt[i]; j++){

printf(" ");

}

if(tt[i] > 9) {

printf("\b"); // backspace : remove 1 space

}

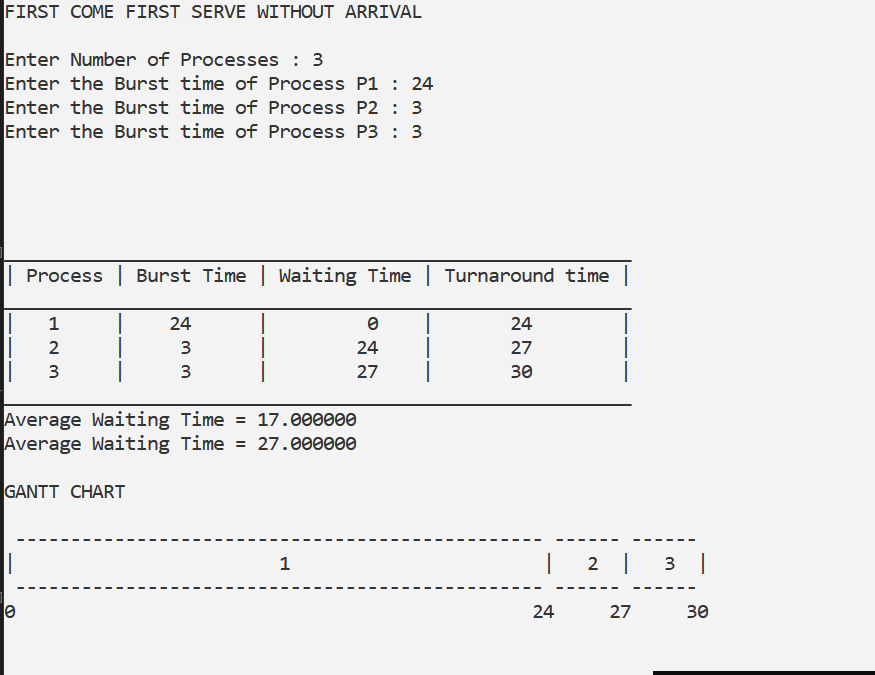
printf("%d", tt[i]);

}

printf("\n");

return 0;

}



2]

//Programmer Name: Sharvil Prabhudesai 20co41

//Program Title : To implement First come first serve with arrival time.

#include <stdio.h>

#include <stdlib.h>

int process[20];

int priority[20];

int bt[20];

int at[20];

int wt[20];

int tt[20];

int n;

void fcfsArrival();

void sort();

void ganttChart();

void display();

int main()

{

printf("FIRST COME FIRST SERVE WITH ARRIVAL\n\n");

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter following process details:");

int i;

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

{

printf("\nEnter arrival time of P%d : ", i + 1);

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

printf("Enter burst time of P%d : ", i + 1);

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

process[i] = i;

}

sort();

fcfsArrival();

ganttChart();

display();

return 0;

}

void fcfsArrival()

{

int i;

int time = 0;

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

{

if (at[process[i]] <= time)

{

wt[process[i]] = time - at[process[i]];

tt[process[i]] = wt[process[i]] + bt[process[i]];

time = time + bt[process[i]];

}

else

{

wt[process[i]] = 0;

tt[process[i]] = bt[process[i]];

time = at[process[i]] + bt[process[i]];

}

}

}

void sort()

{

int j;

int i;

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

{

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

{

if (at[process[j]] < at[process[j - 1]])

{

int temp = process[j];

process[j] = process[j - 1];

process[j - 1] = temp;

}

}

}

}

void ganttChart()

{

printf("\nGantt chart is as follows:\n\n|");

int i;

int time = 0;

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

{

if (at[process[i]] <= time)

{

printf(" P%d |", process[i] + 1);

time = time + bt[process[i]];

}

else

{

printf(" - |");

time = at[process[i]];

i--;

}

}

printf("\n");

time = 0;

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

{

if (at[process[i]] <= time)

{

printf("%d ", time);

time = time + bt[process[i]];

}

else

{

printf("%d ", time);

time = at[process[i]];

i--;

}

}

printf("%d\n\n", time);

}

void display()

{

int i;

float awt = 0;

float att = 0;

printf("The details of processes are as follows:\n\n");

printf("Process Arrival Time Burst Time Waiting Time Turnaround Time\n");

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

{

printf("P%d \t\t%d \t\t%d \t\t%d\t %d\n", i + 1, at[i], bt[i], wt[i], tt[i]);

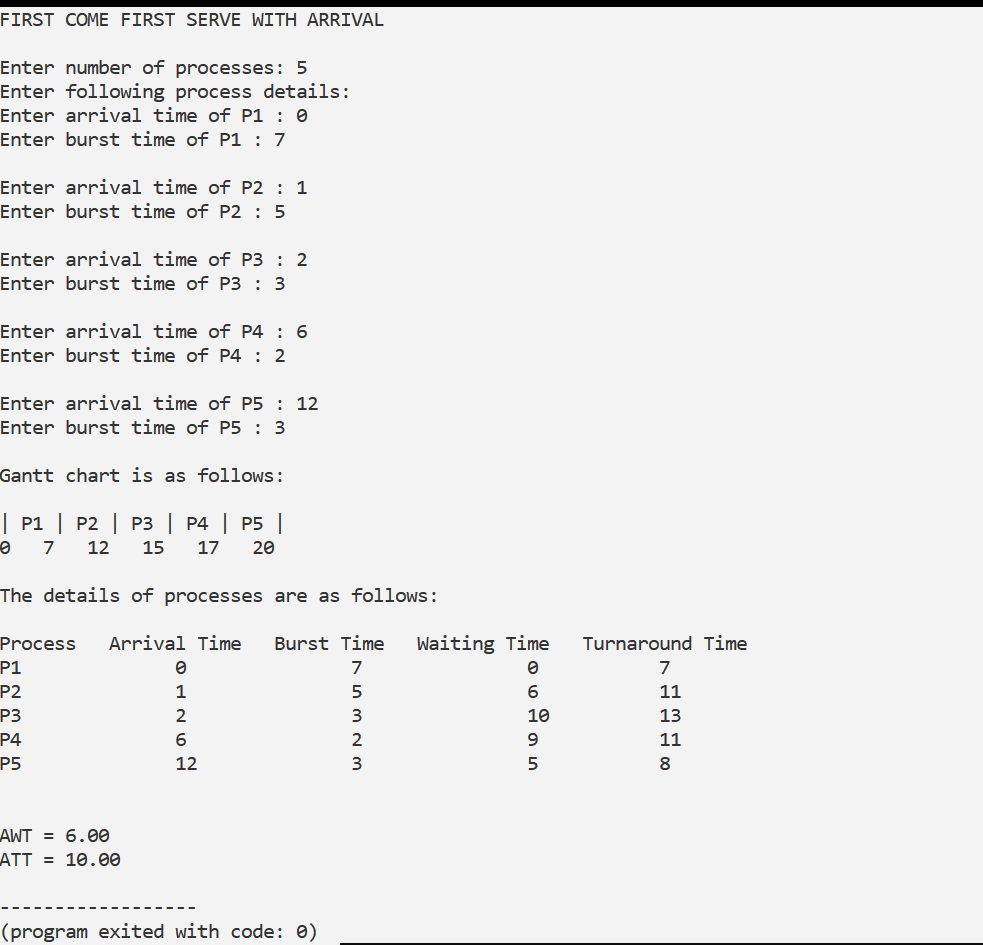
awt = awt + wt[i];

att = att + tt[i];

}

printf("\n\nAWT = %0.2f \nATT = %0.2f", (awt / n), (att / n));

}



3]

//Programmer Name: Sharvil Prabhudesai 20co41

//Program Title : To implement Shortest Job first without arrival time.

#include<stdio.h>

int main ()

{

int n,i,j,l,t;

printf("SHORTEST JOB FIRST WITHOUT ARRIVAL\n\n");

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

scanf("%d",&n);

int bt[n],wt[n],tt[n];

float awt=0,att=0;

int btt;

int p[n];

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

{ p[i]=i+1;

printf("enter the burst time for Process P%d : ",p[i]);

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

btt=btt+bt[i];

}

//using bubbble sort

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

{

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

{ if (bt[j] >bt[j + 1])

{ t=bt[j];

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

bt[j+1]=t;

l=p[j];

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

p[j+1]=l;

}

}

}

wt[0]=0,tt[0]=bt[0];

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

{

wt[i]=wt[i-1]+bt[i-1];

awt=awt+wt[i];

}

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

{

tt[i]=tt[i-1]+bt[i];

att=att+tt[i];

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Process\t | Burst time\t | waiting time\t | turnaround time|\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

{

printf("%d\t | %d\t\t | %d\t\t | %d\t\t |\n",p[i],bt[i],wt[i],tt[i]);

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\nAverage Waiting time = %f\n",(awt/n));

printf("\nAverage Turnaround time = %f",(att+tt[0])/n);

printf("\n\nGANTT CHART ");

printf("\n\n");

// print top bar

printf(" ");

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

for(j=0; j<bt[i]; j++){

printf("--");

}

printf(" ");

}

printf("\n|");

// printing process id in the middle

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

for(j=0; j<bt[i]- 1; j++){

printf(" ");

}

printf(" %d", p[i]);

for(j=0; j<bt[i] - 1; j++) {

printf(" ");

}

printf("|");

}

printf("\n ");

// printing bottom bar

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

for(j=0; j<bt[i]; j++) {

printf("--");

}

printf(" ");

}

printf("\n");

// printing the time line

printf("0");

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

for(j=0; j<bt[i]; j++){

printf(" ");

}

if(tt[i] > 9) {

printf("\b"); // backspace : remove 1 space

}

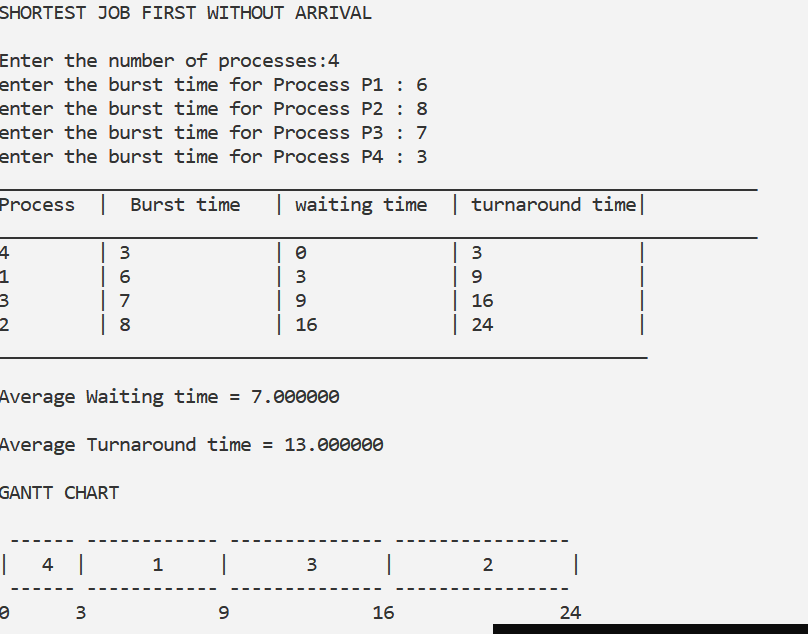
printf("%d", tt[i]);

}

printf("\n");

return 0;

}



4]

//Programmer Name: Sharvil Prabhudesai 20co41

//Program Title : To implement Shortest Job first with arrival time.

#include <stdio.h>

#include <stdlib.h>

int process[20];

int priority[20];

int bt[20];

int at[20];

int wt[20];

int tt[20];

int n;

void shortest\_job\_first\_Arrival();

void sort();

void ganttChart();

void display();

int main()

{

printf("SHORTEST JOB FIRST WITH ARRIVAL\n\n");

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter following process details:");

int i;

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

{

printf("\nEnter arrival time of P%d : ", i + 1);

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

printf("Enter burst time of P%d : ", i + 1);

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

process[i] = i;

}

sort();

shortest\_job\_first\_Arrival();

ganttChart();

display();

return 0;

}

void shortest\_job\_first\_Arrival()

{

int time = 0;

int i;

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

{

int j = i;

int k = i;

if (at[process[i]] > time)

{

while (j < n && at[process[j]] == at[process[k]])

{

if (bt[process[j]] <= bt[process[k]])

{

k = j;

}

j++;

}

}

else

{

while (j < n && at[process[j]] <= time)

{

if (bt[process[j]] < bt[process[k]])

{

k = j;

}

j++;

}

}

int temp = process[k];

process[k] = process[i];

process[i] = temp;

if (at[process[i]] <= time)

{

wt[process[i]] = time - at[process[i]];

tt[process[i]] = wt[process[i]] + bt[process[i]];

time = time + bt[process[i]];

}

else

{

wt[process[i]] = 0;

tt[process[i]] = bt[process[i]];

time = at[process[i]] + bt[process[i]];

}

}

}

void sort()

{

int j;

int i;

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

{

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

{

if (at[process[j]] < at[process[j - 1]])

{

int temp = process[j];

process[j] = process[j - 1];

process[j - 1] = temp;

}

}

}

}

void ganttChart()

{

printf("\nGantt chart is as follows:\n\n|");

int i;

int time = 0;

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

{

if (at[process[i]] <= time)

{

printf(" P%d |", process[i] + 1);

time = time + bt[process[i]];

}

else

{

printf(" - |");

time = at[process[i]];

i--;

}

}

printf("\n");

time = 0;

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

{

if (at[process[i]] <= time)

{

printf("%d ", time);

time = time + bt[process[i]];

}

else

{

printf("%d ", time);

time = at[process[i]];

i--;

}

}

printf("%d\n\n", time);

}

void display()

{

int i;

float awt = 0;

float att = 0;

printf("The details of processes are as follows:\n\n");

printf("Process Arrival Time Burst Time Waiting Time Turnaround Time\n");

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

{

printf("P%d \t\t%d \t\t%d \t\t%d\t %d\n", i + 1, at[i], bt[i], wt[i], tt[i]);

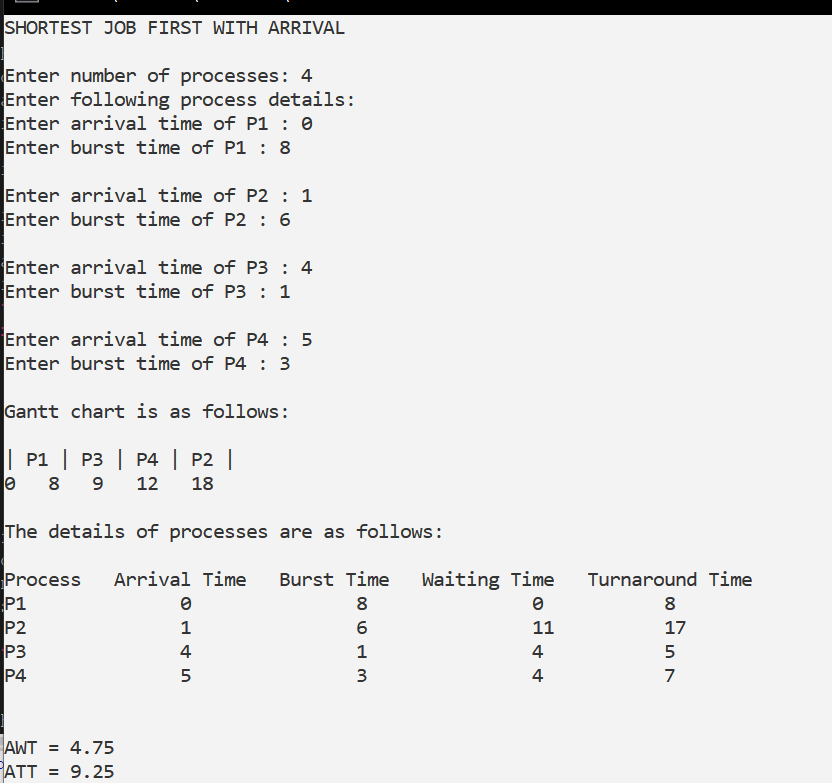
awt = awt + wt[i];

att = att + tt[i];

}

printf("\n\nAWT = %0.2f \nATT = %0.2f", (awt / n), (att / n));

}



5]

//Programmer Name: Sharvil Prabhudesai 20co41

//Program Title : To implement Non premptive priority without arrival time.

#include<stdio.h>

int main ()

{

int n,i,j,l,t,temp;

printf("NON-PREMTIVE PRIORITY SCHEDULING WITHOUT ARRIVAL\n\n");

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

scanf("%d",&n);

int bt[n],wt[n],tt[n];

float awt=0,att=0;

int btt;

int p[n];

int priority[n];

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

{ p[i]=i+1;

printf("enter the burst time for process P%d : ",p[i]);

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

printf("enter the priority for process P%d : ",p[i]);

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

btt=btt+bt[i];

}

//using bubbble sort

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

{

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

{ if (priority[j] >priority[j + 1])

{ t=priority[j];

priority[j]=priority[j+1];

priority[j+1]=t;

temp=bt[j];

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

bt[j+1]=temp;

l=p[j];

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

p[j+1]=l;

}

}

}

wt[0]=0,tt[0]=bt[0];

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

{

wt[i]=wt[i-1]+bt[i-1];

awt=awt+wt[i];

}

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

{

tt[i]=tt[i-1]+bt[i];

att=att+tt[i];

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Process\t | Burst time\t | waiting time\t | turnaround time|\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

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

{

printf("P%d\t | %d\t\t | %d\t\t | %d\t\t |\n",p[i],bt[i],wt[i],tt[i]);

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\nAverage Waiting time = %f\n",(awt/n));

printf("\nAverage Turnaround time = %f",(att+tt[0])/n);

printf("\n\nGANTT CHART ");

printf("\n\n");

// print top bar

printf(" ");

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

for(j=0; j<bt[i]; j++){

printf("--");

}

printf(" ");

}

printf("\n|");

// printing process id in the middle

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

for(j=0; j<bt[i]- 1; j++){

printf(" ");

}

printf("P%d", p[i]);

for(j=0; j<bt[i] - 1; j++) {

printf(" ");

}

printf("|");

}

printf("\n ");

// printing bottom bar

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

for(j=0; j<bt[i]; j++) {

printf("--");

}

printf(" ");

}

printf("\n");

// printing the time line

printf("0");

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

for(j=0; j<bt[i]; j++){

printf(" ");

}

if(tt[i] > 9) {

printf("\b"); // backspace : remove 1 space

}

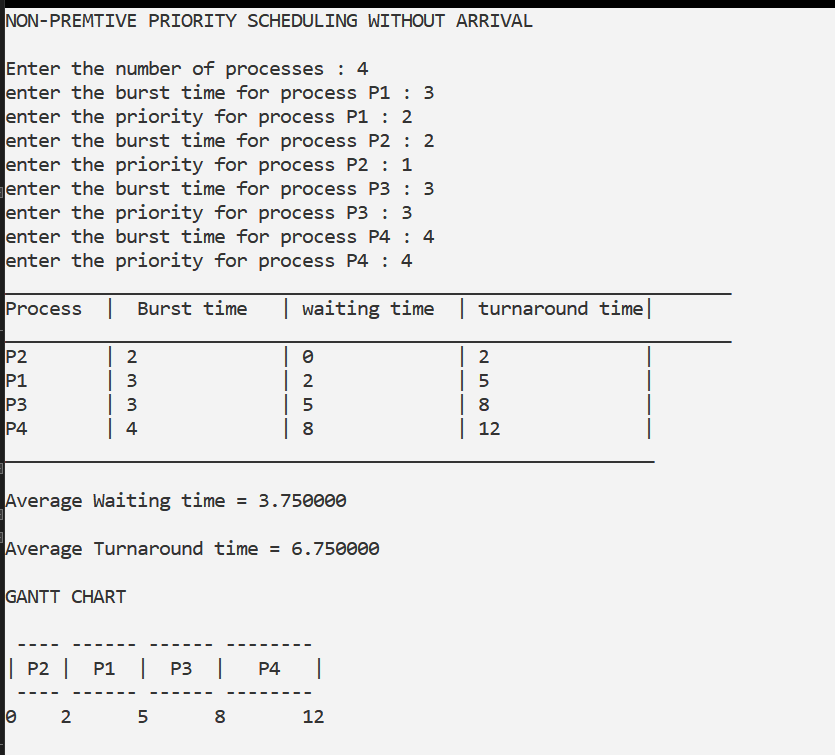
printf("%d", tt[i]);

}

printf("\n");

return 0;

}



6]

