**Scheduling Algorithm**

**FCFS**

**PROGRAM**

#include<stdio.h>

int main()

{

int n;

float awt,twt=0.0,ttat=0.0,attat;

printf("enter the number of proces");

scanf("%d",&n);

int bt[n],p[n],a[n],b[n];

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

{

printf(" enter the burst time of the process P %d \n",i);

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

}

printf(" The burst time table \n");

printf(" Process \t Burst Time \n");

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

{

printf(" %d \t\t %d \n",i,bt[i]);

}

//awt

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

{

a[0]=0;

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

}

printf("Waiting Time Table \n");

printf(" Process \t Waiting Time \n");

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

{

//p[i]=a[i];

printf(" %d \t\t %d \n",i,a[i]);

}

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

{

twt=twt+a[i];

}

awt=twt/n;

printf("Total Waiting Time %f \n",twt);

printf("Average Waiting Time %f \n",awt);

//atat

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

{

b[i]=bt[i] + a[i];

}

printf("Turn Around Time Table \n");

printf(" Process \t Tat Time \n");

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

{

printf(" %d \t\t %d \n",i,b[i]);

}

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

{

ttat=ttat+b[i];

}

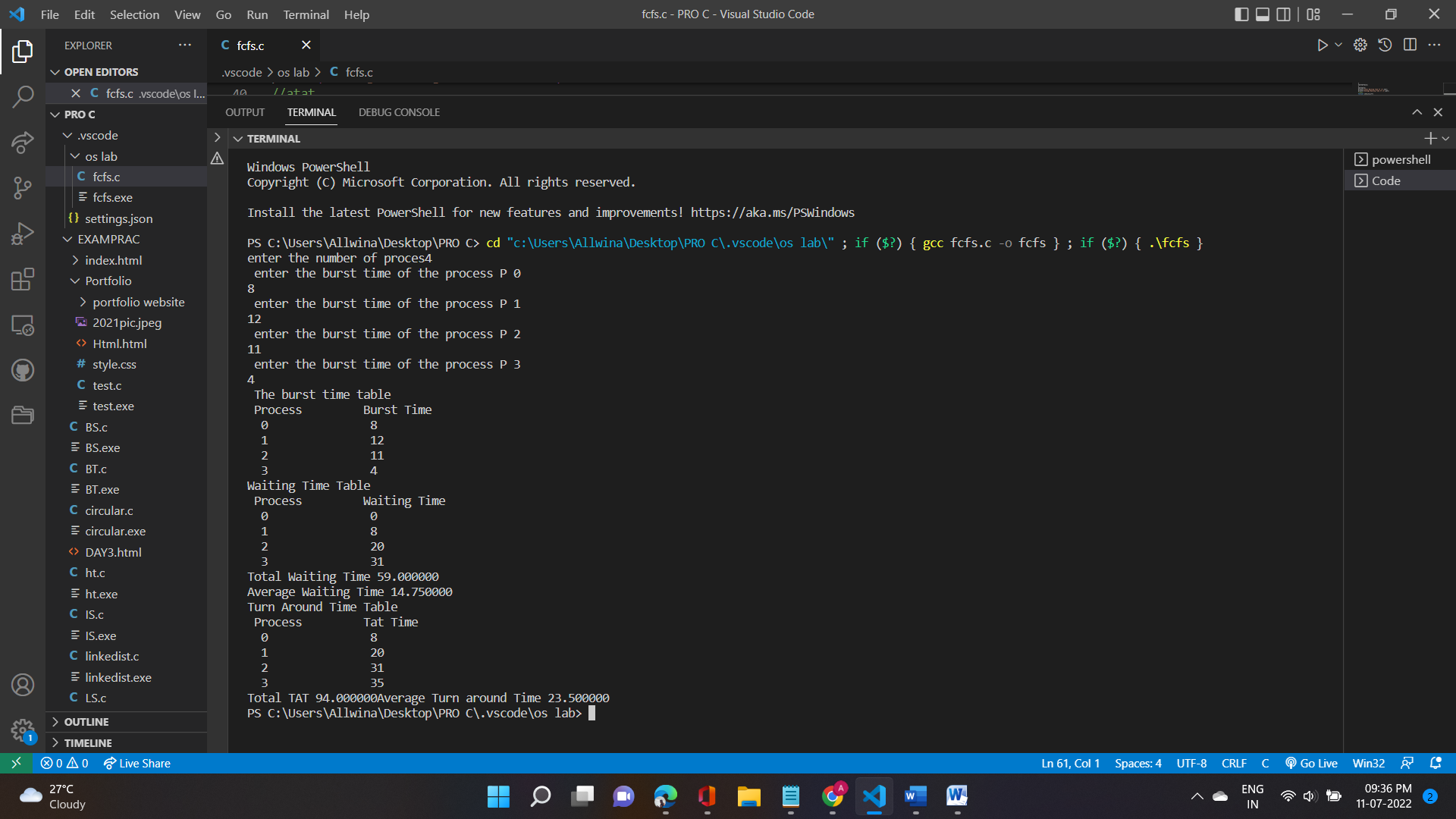
printf("Total TAT %f",ttat);

attat=ttat/n;

printf("Average Turn around Time %f",attat);

}

**OUTPUT**



**SJF**

**PROGRAM**

#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("\n Enter 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("\n Processt Burst Time \t Waiting Time \tTurnaround 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 Average Waiting Time=%f",avg\_wt);

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

// printf("\n |");

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

{

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

}

printf("\n");

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

{

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

}

printf("|\n");

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

{

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

}

printf(" \n");

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

{

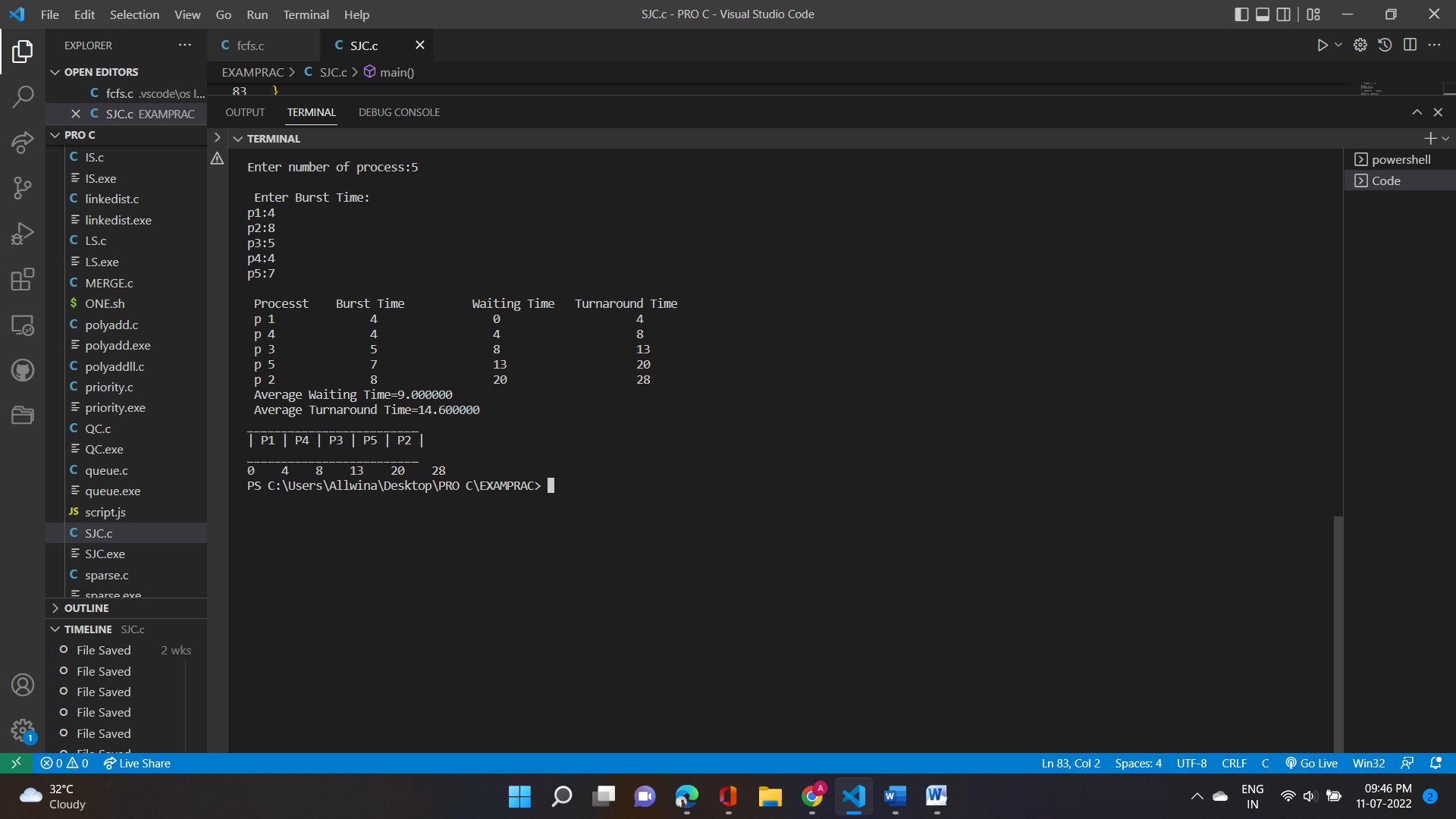
printf("%d ",wt[i]);

}

printf("%d ",tat[n-1]);

}

**OUTPUT**



**PRIORITY SCHEDULING**

**PROGRAM**

#include<stdio.h>

int main()

{

int n,temp=0,min;

float awt,twt=0.0,ttat=0.0,attat;

printf("enter the number of processses");

scanf("%d",&n);

int pr[n],bt[n],p[n],a[n],b[n];

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

{

printf("enter the process ");

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

printf("enter the burst time ");

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

printf("enter the priority");

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

}

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

{

min = i;

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

{

if (pr[j] < pr[min])

{

min= j;

}

}

temp = pr[i];

pr[i] = pr[min];

pr[min] = temp;

temp = bt[i];

bt[i] = bt[min];

bt[min] = temp;

temp = p[i];

p[i] = p[min];

p[min] = temp;

}

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

{

a[0]=0;

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

}

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

{

twt=twt+a[i];

}

awt=twt/n;

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

{

b[i]=bt[i]+a[i];

}

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

{

ttat=ttat+b[i];

}

attat=ttat/n;

printf("Process \t priority \t burst time \t waiting time \t Turn Around Time\n");

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

{

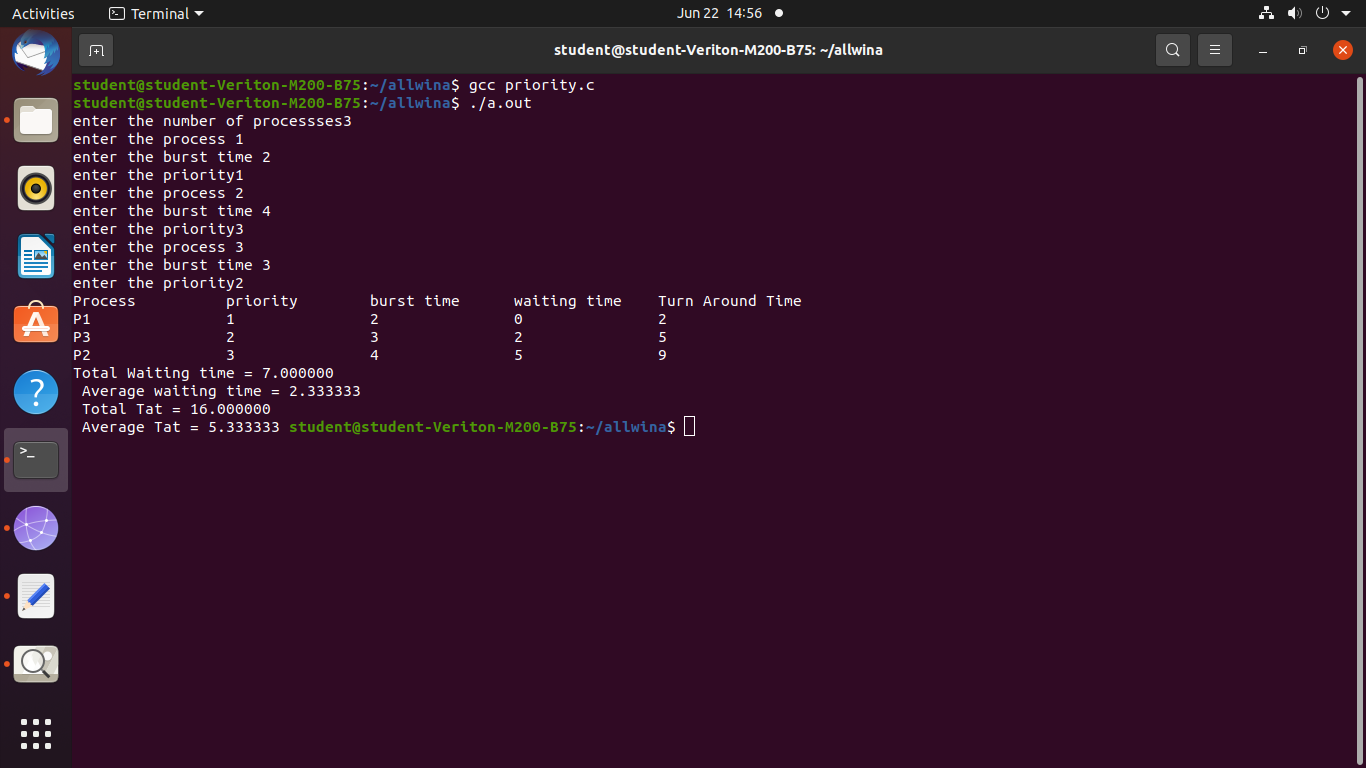
printf("P%d \t\t %d \t\t %d \t\t %d \t\t %d \n",p[i],pr[i],bt[i],a[i],b[i]);

}

printf("Total Waiting time = %f \n Average waiting time = %f \n Total Tat = %f \n Average Tat = %f ",twt,awt,ttat,attat);

}

**OUTPUT**



**ROUND ROBIN 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 Timen");

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;

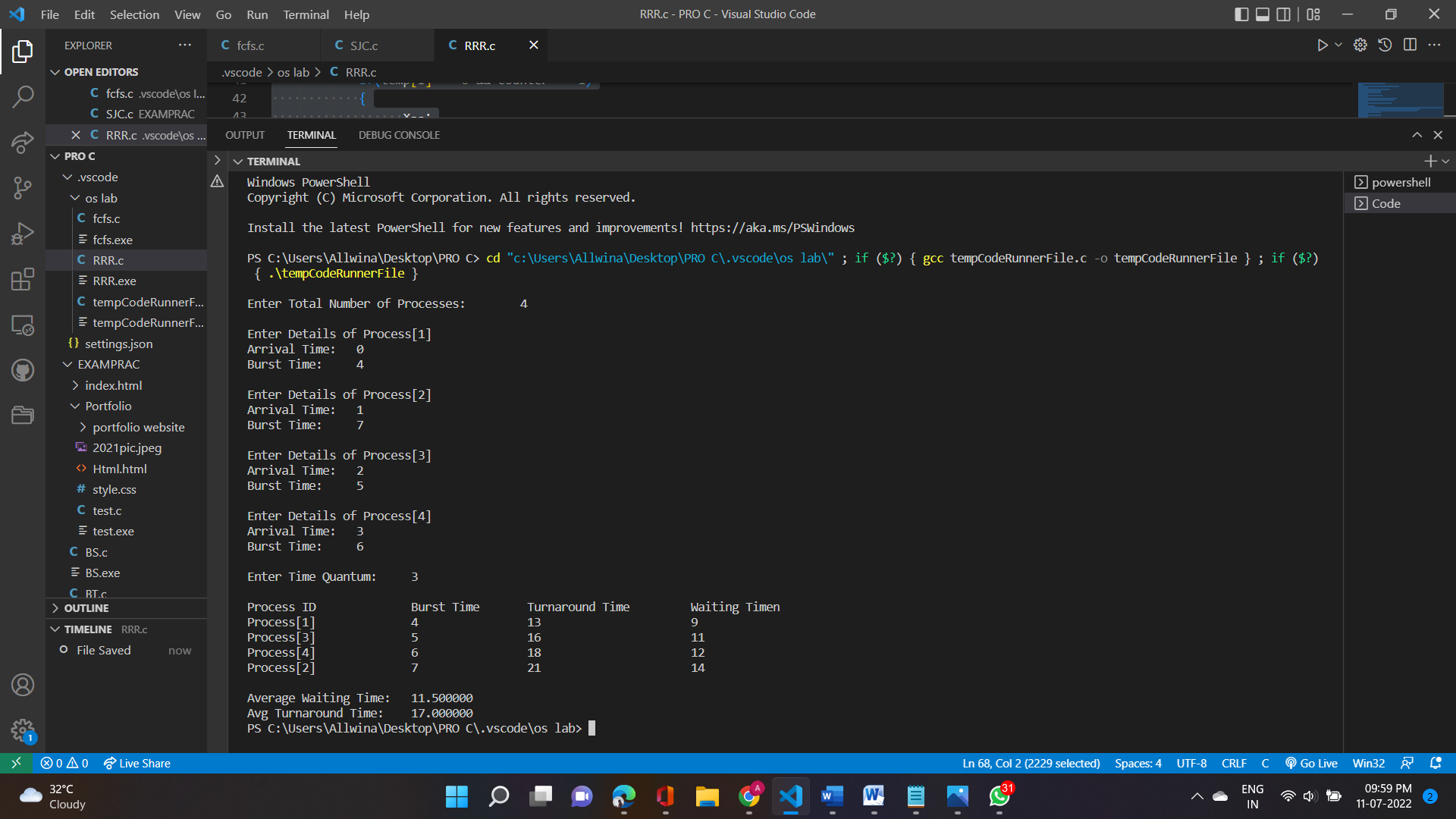
printf("\n\nAverage Waiting Time:\t%f", average\_wait\_time);

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

return 0;

}

**OUTPUT**



**PRODUCER-CONSUMER PROBLEM**

**PROGRAM**

#include<stdio.h>

#include<stdlib.h>

int mutex=1;

int full=0;

int empty=5,x=0;

void producer()

{

--mutex;

++full;

--empty;

x++;

printf("\nProducer produces " "item %d",x);

++mutex;

}

void consumer()

{

--mutex;

--full;

++empty;

printf("\nConsumer consumes" "item %d",x);

x--;

++mutex;

}

int main()

{

int n,i;

printf("\n 1.Press 1 for Producer""\n 2.Press 2 for consumer ""\n3.Press 3 for exit");

for(i=1;i>0;i++)

{

printf("\n Enter your choice:");

scanf("%d",&n);

switch(n)

{

case 1:

if((mutex==1)&&(empty!=0))

{

producer();

}else

{

printf("Buffer is full!");

}

break;

case 2:

if((mutex==1)&&(full!=0))

{

consumer();

}

else

{

printf("Buffer is empty");

}

break;

case 3:

exit(0);

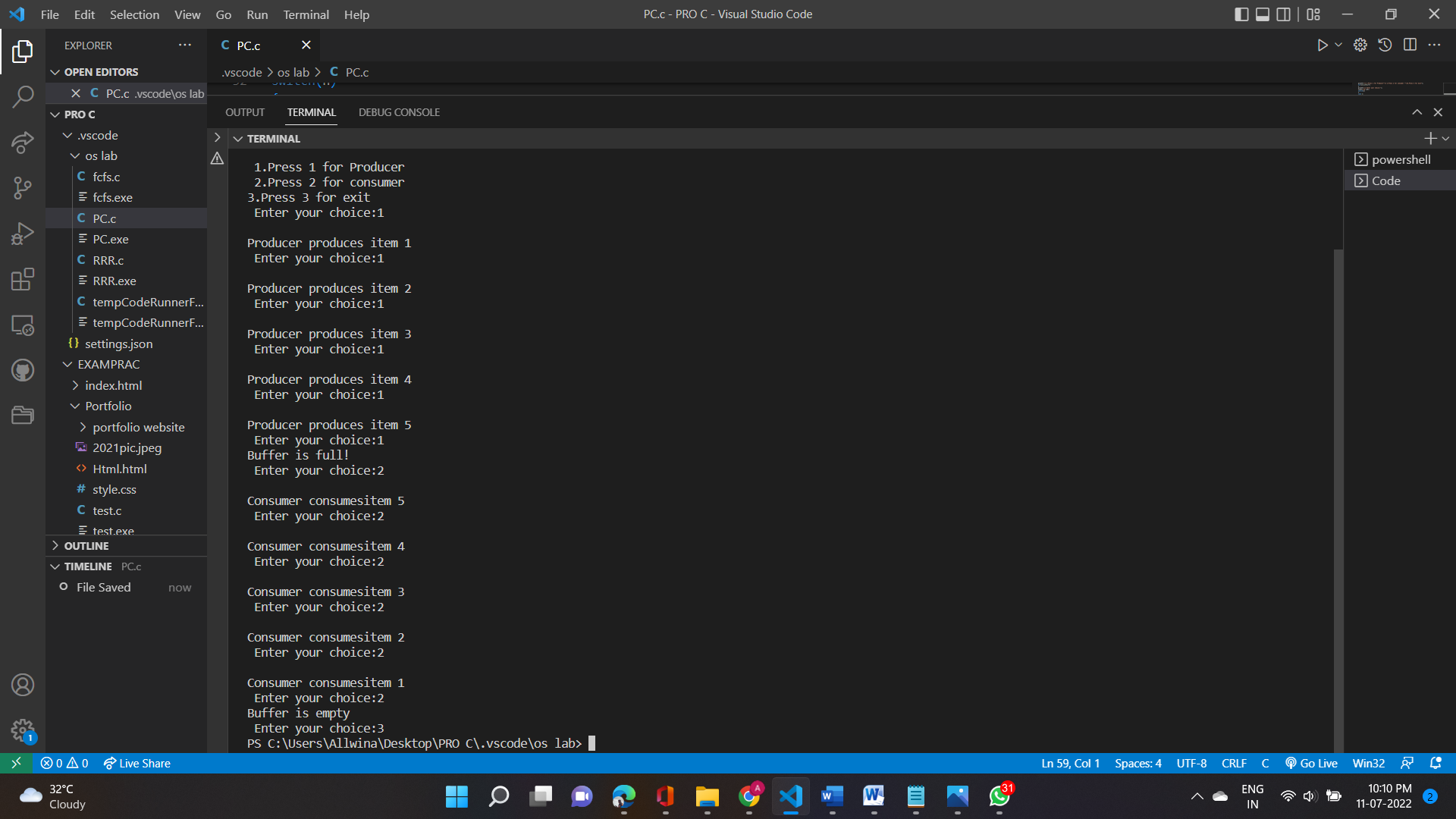
break;

}

}

}

**OUTPUT**



**BANKER’S ALGORITHM**

**PROGRAM**

#include <stdio.h>

int main()

{

int n, m, i, j, k, y,alloc[20][20],max[20][20],avail[50],ind=0;

printf("Enter the no of Proceses:");

scanf("%d",&n);

printf("Enter the no of Resources:");

scanf("%d",&m);

printf("Enter the Allocation Matrix:");

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

for (j = 0; j < m; j++)

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

}

printf("Enter the Max Matrix:");

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

for (j = 0; j < m; j++)

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

}

printf("Enter the Available Matrix");

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

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

int finish[n], safesequence[n],work[m],need[n][m];

//calculating NEED matrix

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

for (j = 0; j < m; j++)

need[i][j] = max[i][j] - alloc[i][j];

}

printf("NEED matrix is");

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

{

printf("\n");

for (j = 0; j < m; j++)

printf(" %d ",need[i][j]);

}

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

{

work[i]=avail[i];

}

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

finish[i] = 0;

}

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

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

{

if (finish[i] == 0)

{

int flag = 0;

for (j = 0; j < m; j++)

{

if (need[i][j] > work[j])

{

flag = 1;

break;

}

}

if (flag == 0) {

safesequence[ind++] = i;

for (y = 0; y < m; y++)

work[y] += alloc[i][y];

finish[i] = 1;

}

}

}

}

printf("\nFollowing is the SAFE Sequence\n");

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

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

}

**OUTPUT**

