**EXERCISE-2**

Implement CPU Scheduling Algorithms: **Priority Scheduling**

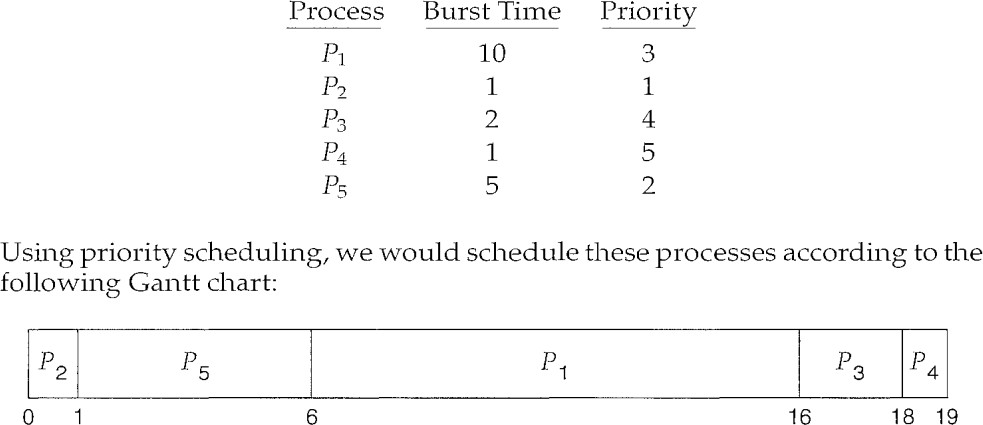
**Aim:**

To Implement CPU Scheduling Algorithms: **Priority Scheduling**.

**DESCRIPTION:**

## Priority CPU scheduling algorithm:

* A priority is associated with each process, and the CPU is allocated to the process with the highest priority.
* Equal-priority processes are scheduled in FCFS order.
* An SJF algorithm is simply a priority algorithm where the priority (p) is the inverse of the (predicted) next CPU burst.
* The larger the CPU burst, the lower the priority, and vice versa.
* Low numbers represent high priority.
* As an example, consider the following set of processes, assumed to have arrived at time 0 in the order P1, P2, · · ·, P5, with the length of the CPU burst given in milliseconds.



* The average waiting time is 8.2 milliseconds.
* A rnajor problem with priority scheduling algorithms is indefinite blocking, or starvation.
* A process that is ready to run but waiting for the CPU can be considered blocked.
* A solution to the problem of indefinite blockage of low-priority processes is aging.
* Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.

Burst Time: The process needs time to complete its execution is called Burst Time.

Arrival Time: The time at which the process arrived is called Arrival time.Generally starts from 0.

Waiting Time: The amount of time the process waited for its execution.

Finish Time: The time at which the process completes its execution.

**Program:**

#include<stdio.h>

struct process

{

char name[10];

int bt,wt,ft,tat,pr;

}p[10],temp;

int main()

{

int i,j,n,ttat=0,twt=0;

float atat,awt;

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

scanf("%d",&n);

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

{

printf("\nEnter the name of the process:");

scanf("%s",p[i].name);

printf("Enter the burst time of the %s process: ",p[i].name);

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

printf("Enter the priority of the %s process :",p[i].name);

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

}

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

{

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

{

if(p[i].pr>p[j].pr)

{

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

printf("Sorted order is:");

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

{

printf(" %s",p[i].name);

}

printf("\n");

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

{

if(i==0)

{

p[i].wt=0;

p[i].ft=p[i].bt;

}

else

{

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

p[i].ft=p[i].wt+p[i].bt;

}

}

printf("\np.name\t bt\t wt\t ft\t tat\n");

printf("----------------------------------------------");

printf("\n");

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

p[i].tat=p[i].ft;

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

{

printf(" %s\t%d\t %d\t %d\t %d\t",p[i].name,p[i].bt,p[i].wt,p[i].ft,p[i].tat);

printf("\n");

}

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

{

printf("\nThe burst time of the process %s is %d",p[i].name,p[i].bt);

printf("\nThe waiting time of the process %s is %d",p[i].name,p[i].wt);

printf("\nThe finish time of the process %s is %d",p[i].name,p[i].ft);

printf("\nThe turn around time of the process %s is %d",p[i].name,p[i].ft);

printf("\n");

}

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

{

ttat=ttat+p[i].ft;

twt=twt+p[i].wt;

}

atat=ttat/n;

awt=twt/n;

printf("\nThe total turn around time is %d",ttat);

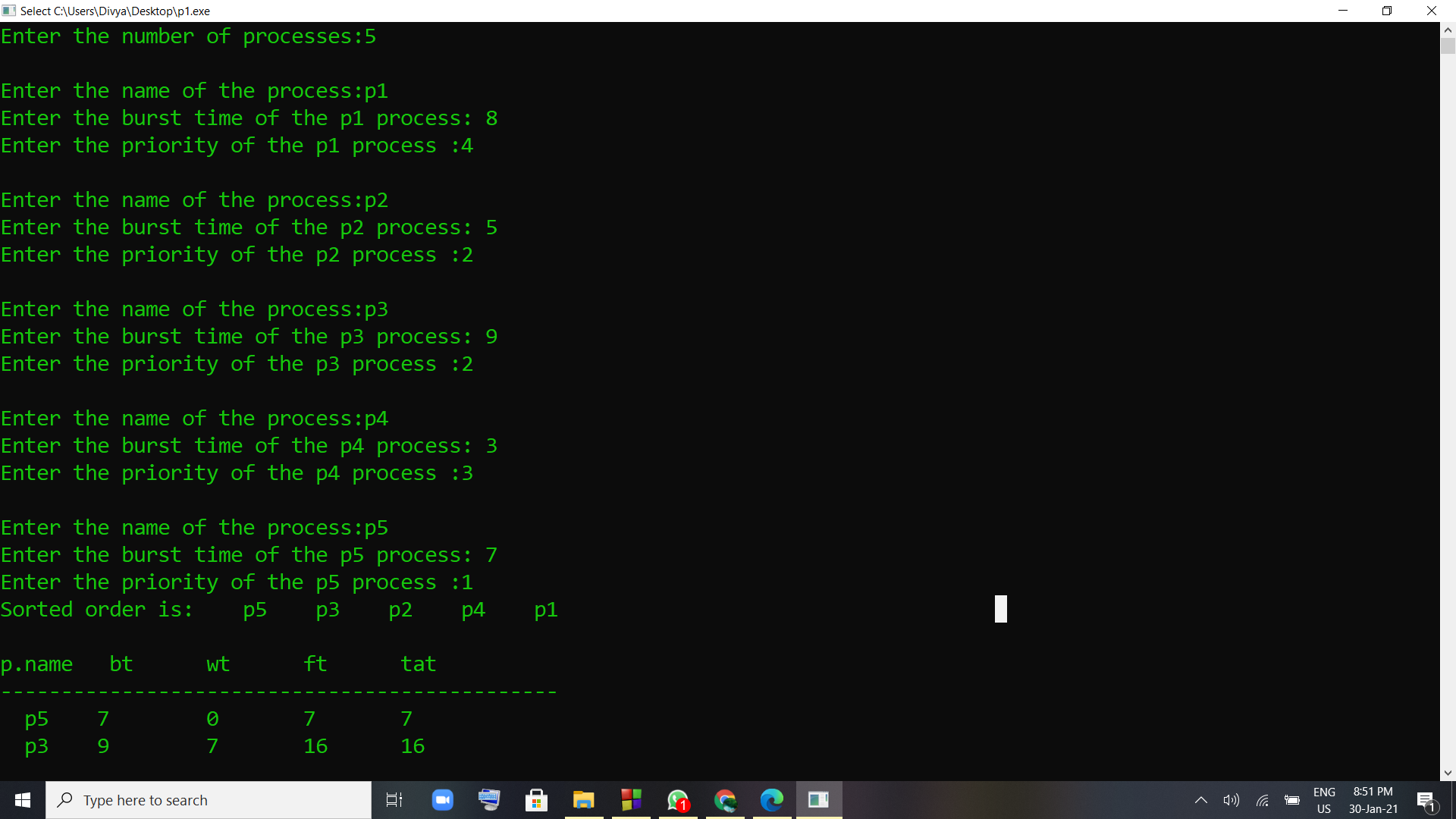
printf("\nThe total waiting time is %d",twt);

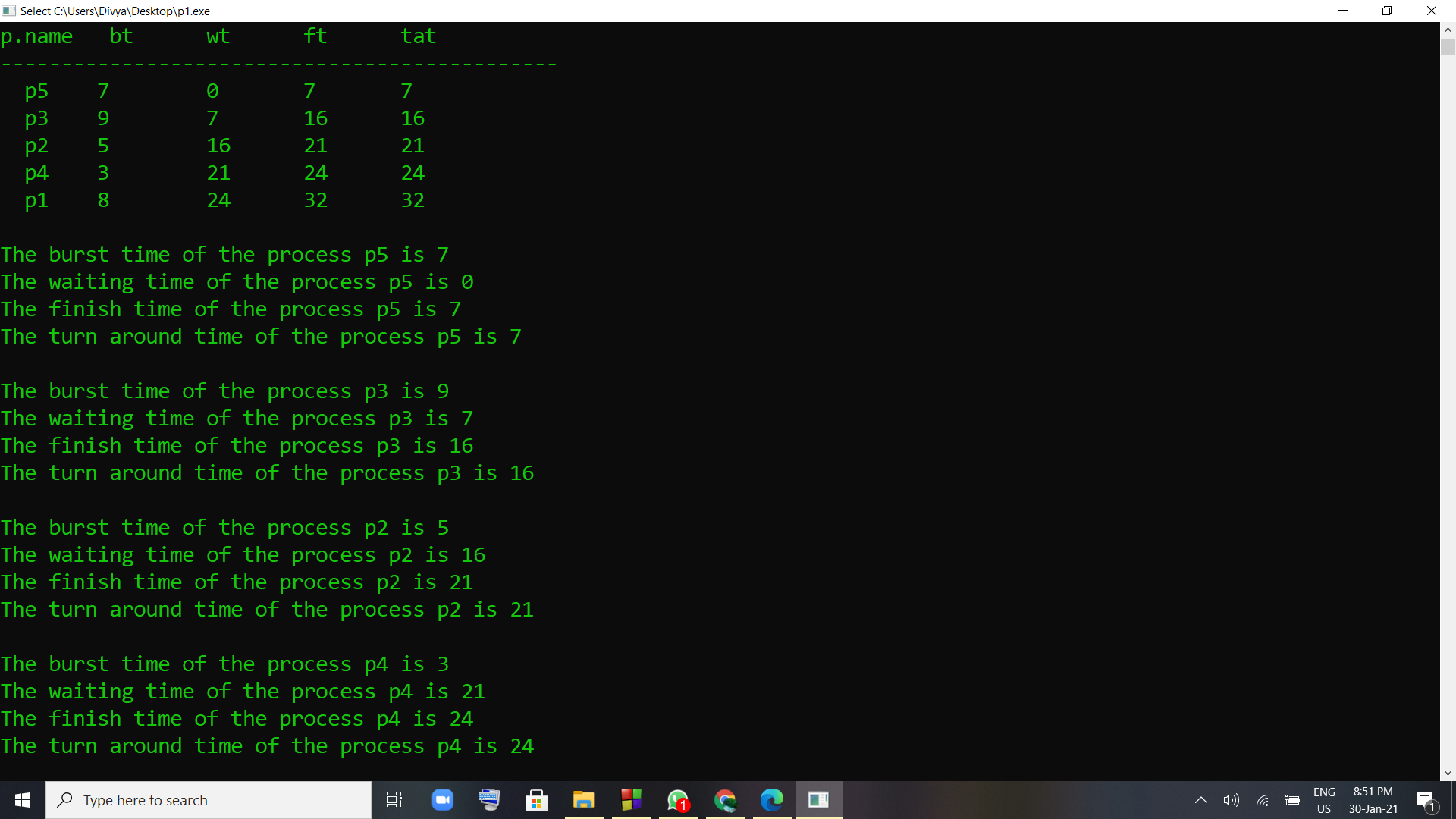
printf("\nThe average turn around time is %f",atat);

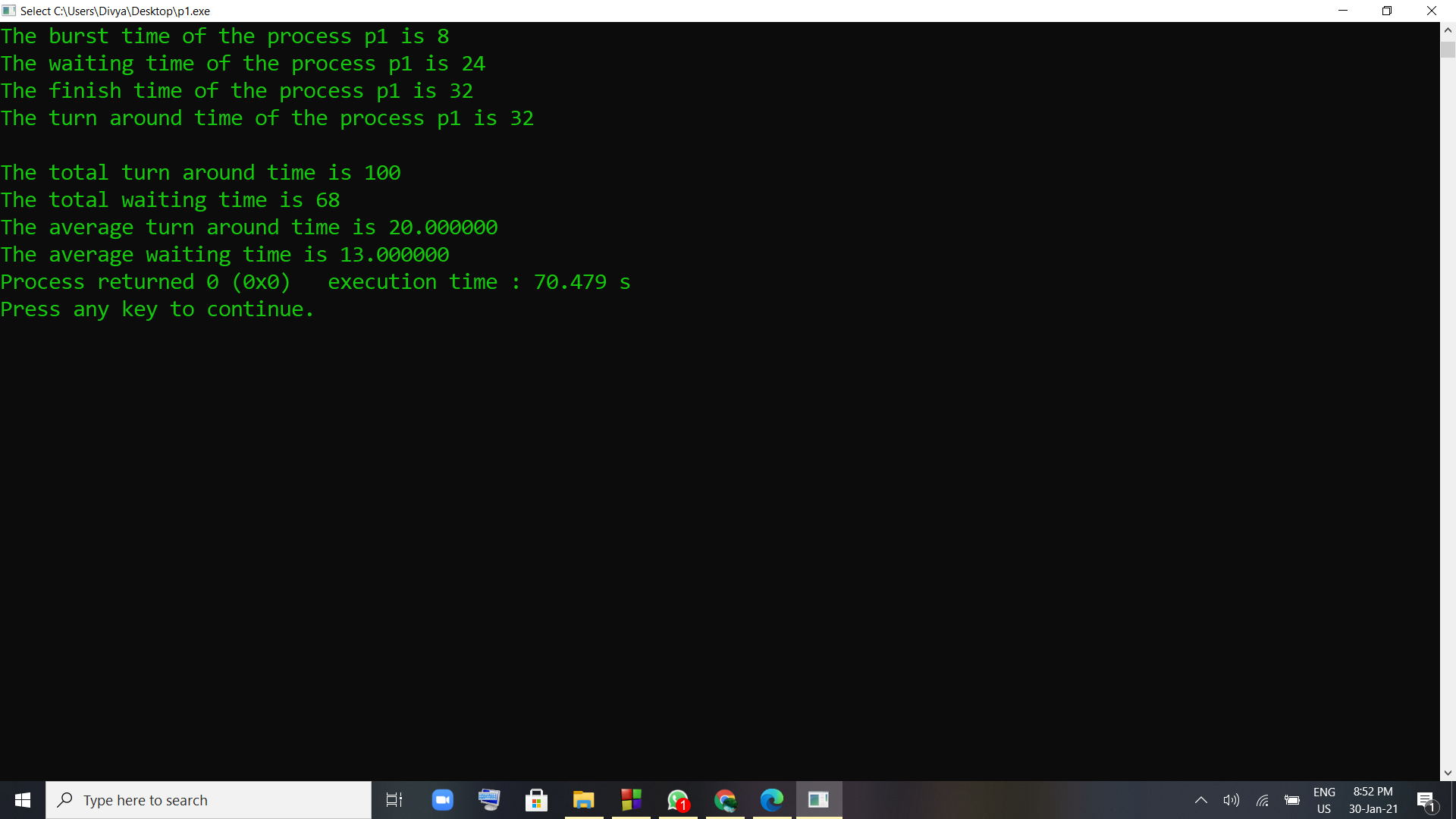
printf("\nThe average waiting time is %f",awt);

}

**Input & Output:**

****

****

****