Programme Name: B.SC. (H)

COMPUTER SCIENCE

SEMESTER-IV

Paper Title:Operating System Practical

Submitted by:

Aditi Varshney

Examination Roll No:19044570040

College Roll No: CSC/19/89

College Name :

Mata Sundri College for Women,University of Delhi

College Address: Mata Sundri Lane,New Delhi 110002

Q1. Write a program (using fork() and/or exec() commands) where parent and child execute:

1. same program, same code.

b) same program, different code.

c) before terminating, the parent waits for the child to finish its task, both for above

mentioned cases a) and b)

Answer:-

A)#include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

using namespace std;

void forkexample()

{

// child process creation

int id1=fork();

cout<<"Hello !"<<getpid()<<"\n";

}

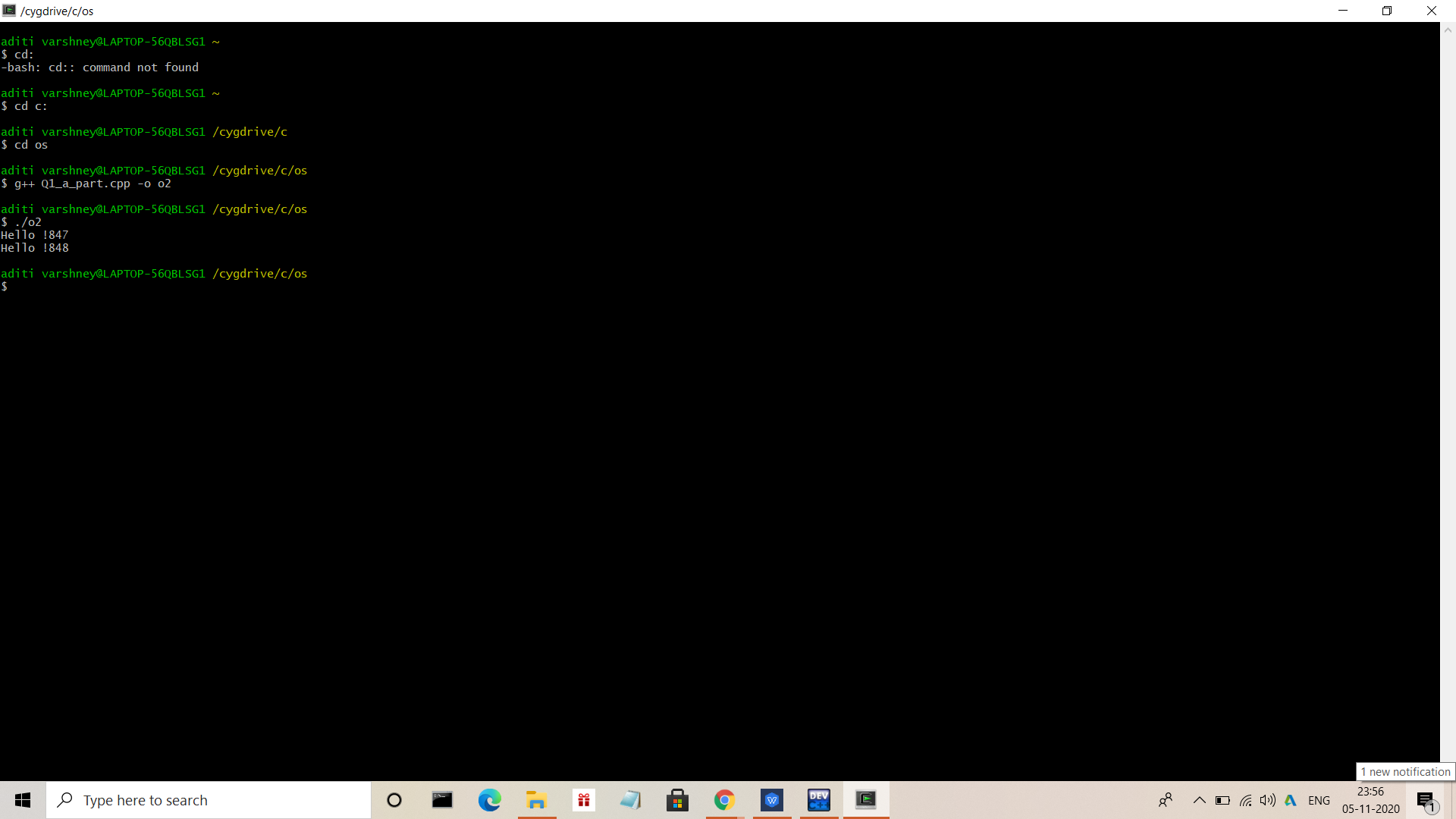
int main()

{

forkexample();

return 0;

}



B)#include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

using namespace std;

void forkexample()

{

// process creation

int id1=fork();

if(id1==0)

cout<<"Hello from Child!"<<getpid()<<endl;

else

cout<<"Hello from Parent !"<<getpid()<<"\n";

}

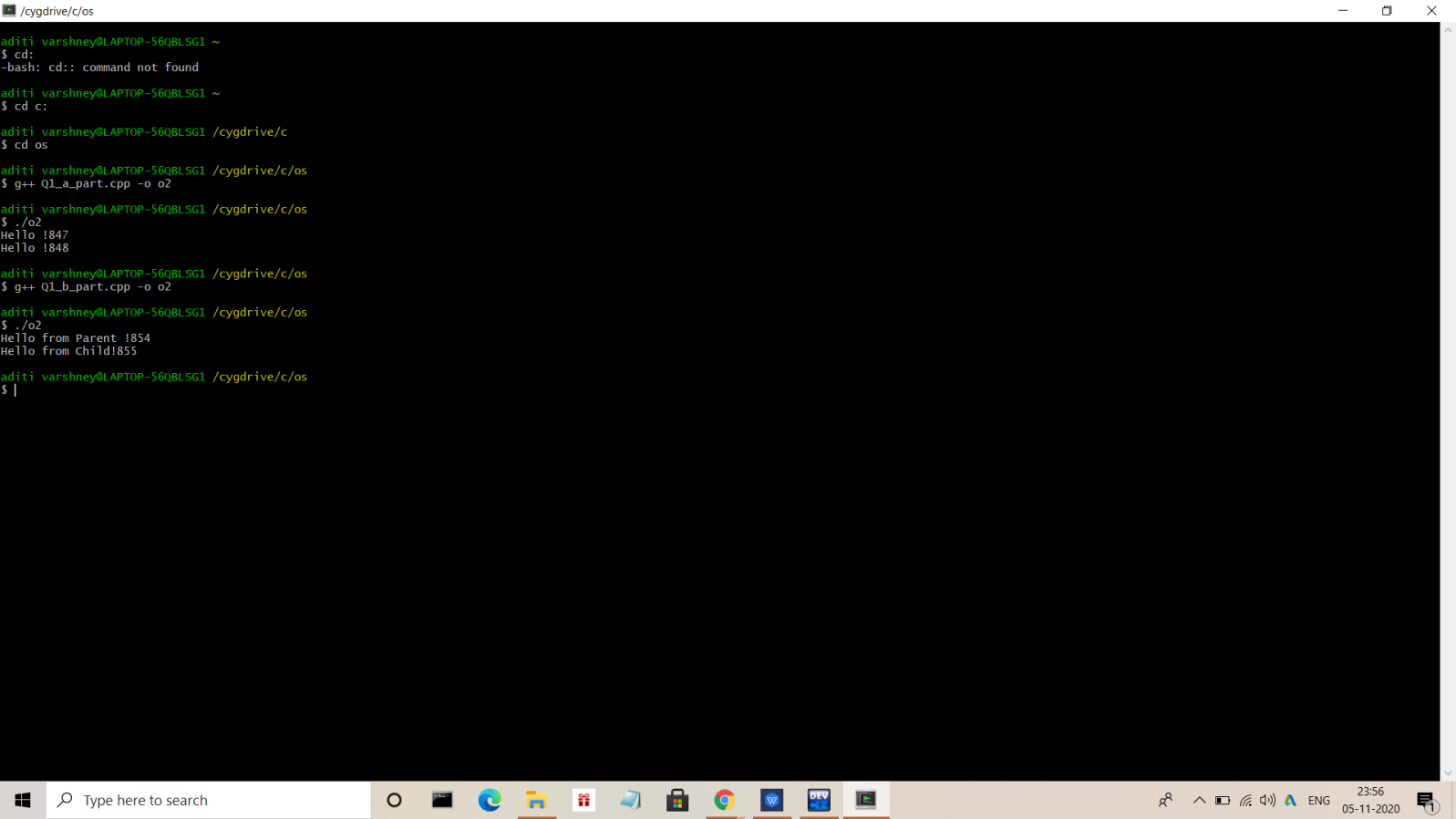
int main()

{

forkexample();

return 0;

}



C)

Case a) #include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

using namespace std;

void forkex()

{

//child process creation

int id1=fork();

if(wait(NULL)!=-1)

cout<<"waiting for child to finish "<<endl;

cout<<"Hello !"<<getpid()<<"\n";

}

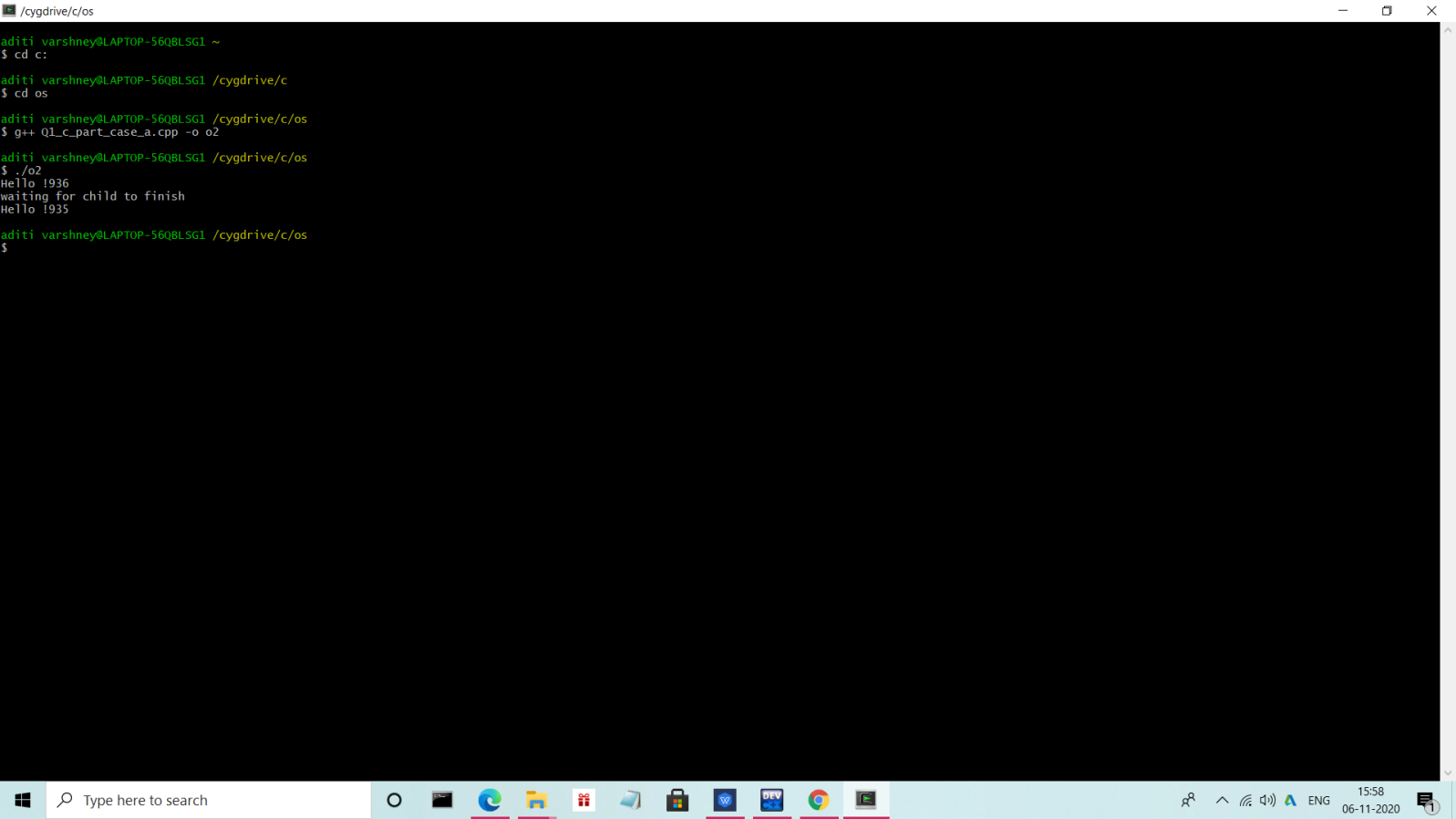
int main()

{

forkex();

return 0;

}



Case b) #include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

using namespace std;

void forkexample()

{

// process creation

int id1=fork();

if(id1==0)

cout<<"Hello from Child!"<<getpid()<<endl;

else{

wait(NULL);

cout<<"Waiting for child to finish!"<<endl;

cout<<"Hello from Parent !"<<getpid()<<"\n";

}

}

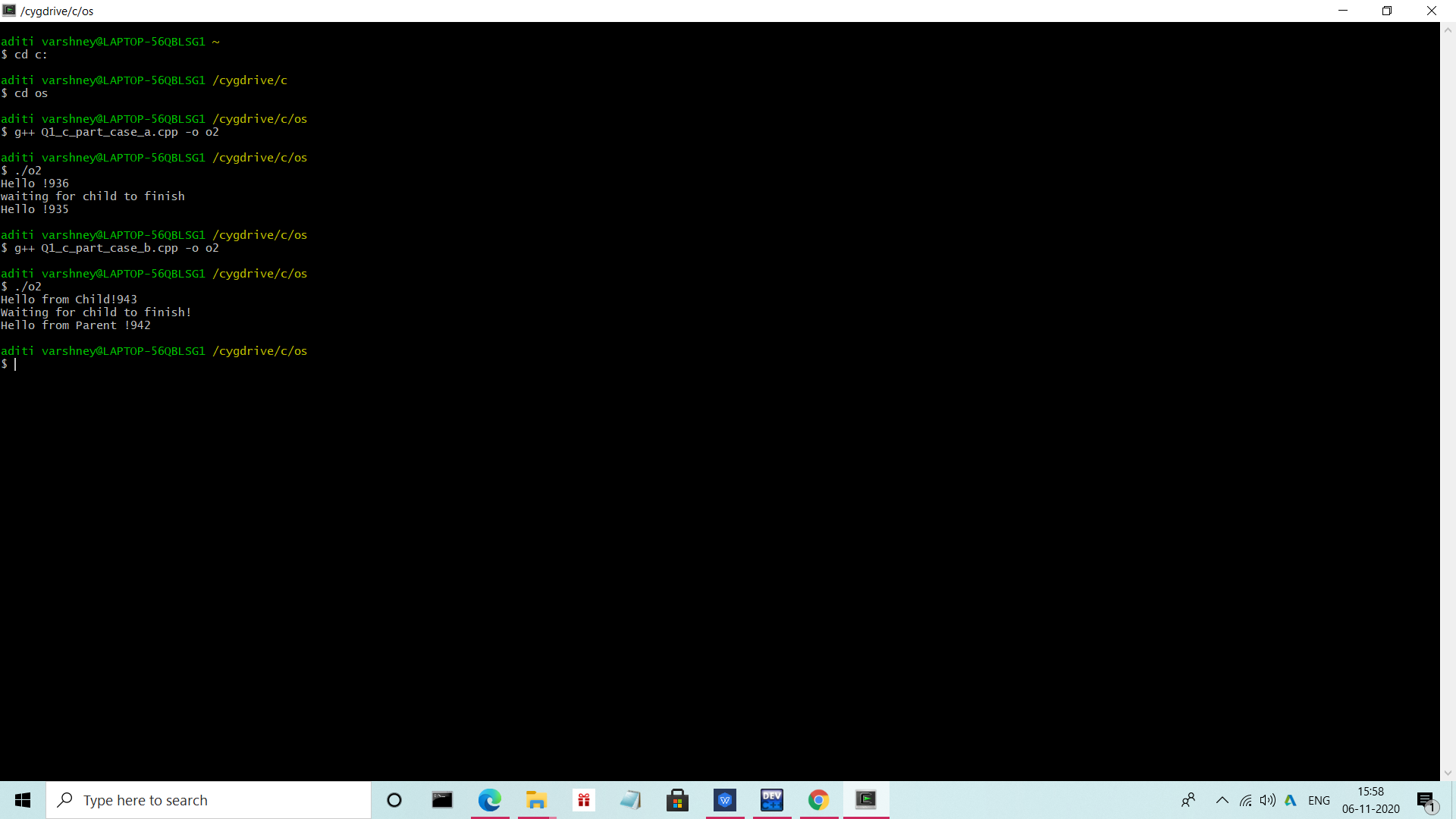
int main()

{

forkexample();

return 0;

}



2. Write a program to show how multiple fork() system calls work.

Answer:-

#include<iostream>

#include<unistd.h>

#include<sys/types.h>

#include<sys/wait.h>

using namespace std;

int main()

{

int ID1=fork();

int ID2=fork();

if(ID1<0)

cout<<"\nUnsuccessful \n";

if(ID1==0)

cout <<" Im child pid = " << getpid() <<" return ID1: "<< ID1<<"\n";

else

cout <<" Im parent pid = " << getpid() <<" return ID1: "<< ID1<<"\n";

if(ID2<0)

cout<<"\nUnsuccessful \n";

if(ID2==0)

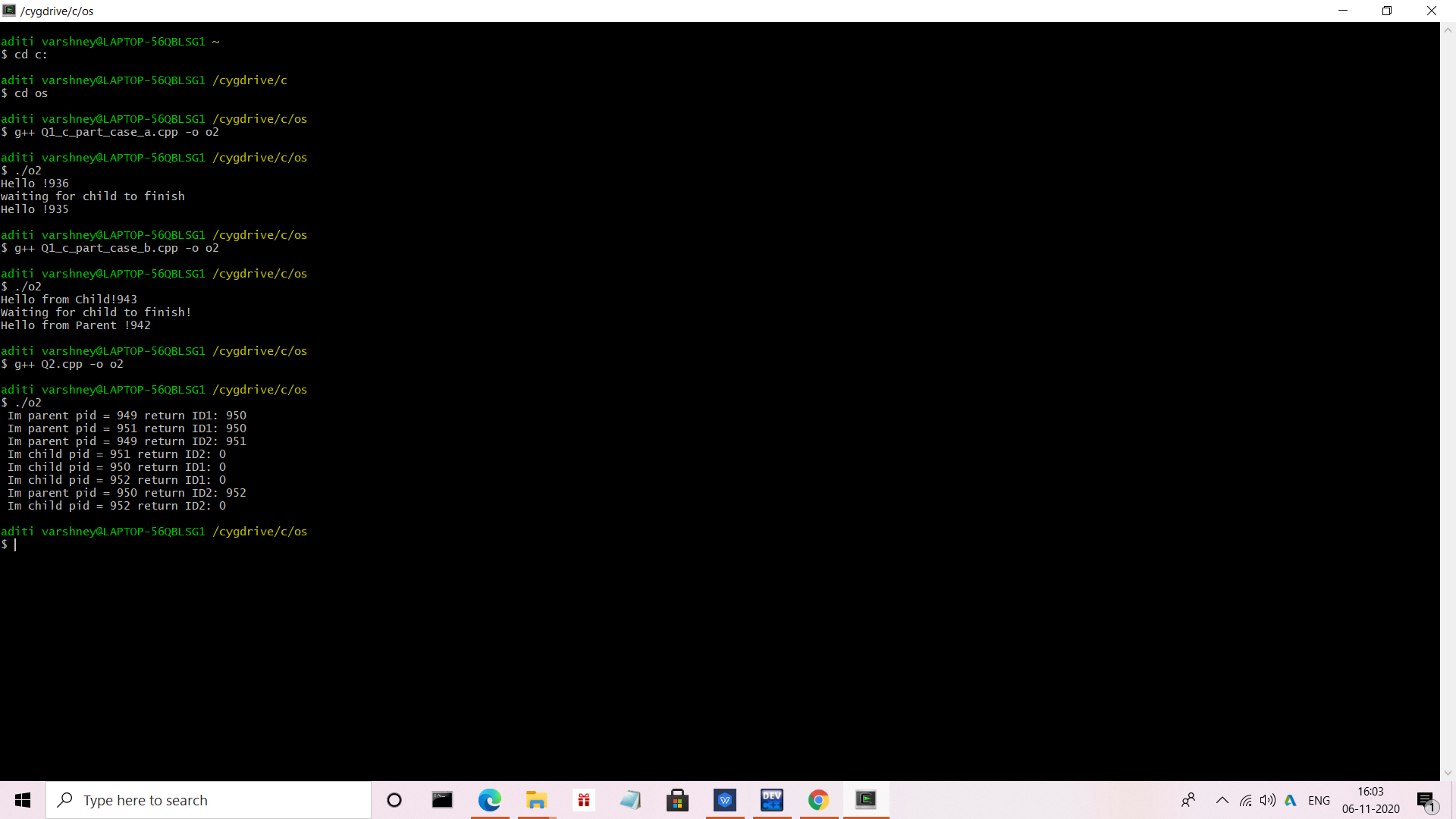
cout <<" Im child pid = " << getpid() <<" return ID2: "<< ID2<<"\n";

else

cout <<" Im parent pid = " << getpid() <<" return ID2: "<< ID2<<"\n";

return 0;

}



3. Write a program to report behaviour of Linux kernel including kernel version, CPU type

and model. (CPU information)

Answer:-

#include<iostream>

using namespace std;

int main()

{

cout<<"\nKernel version is:\n";

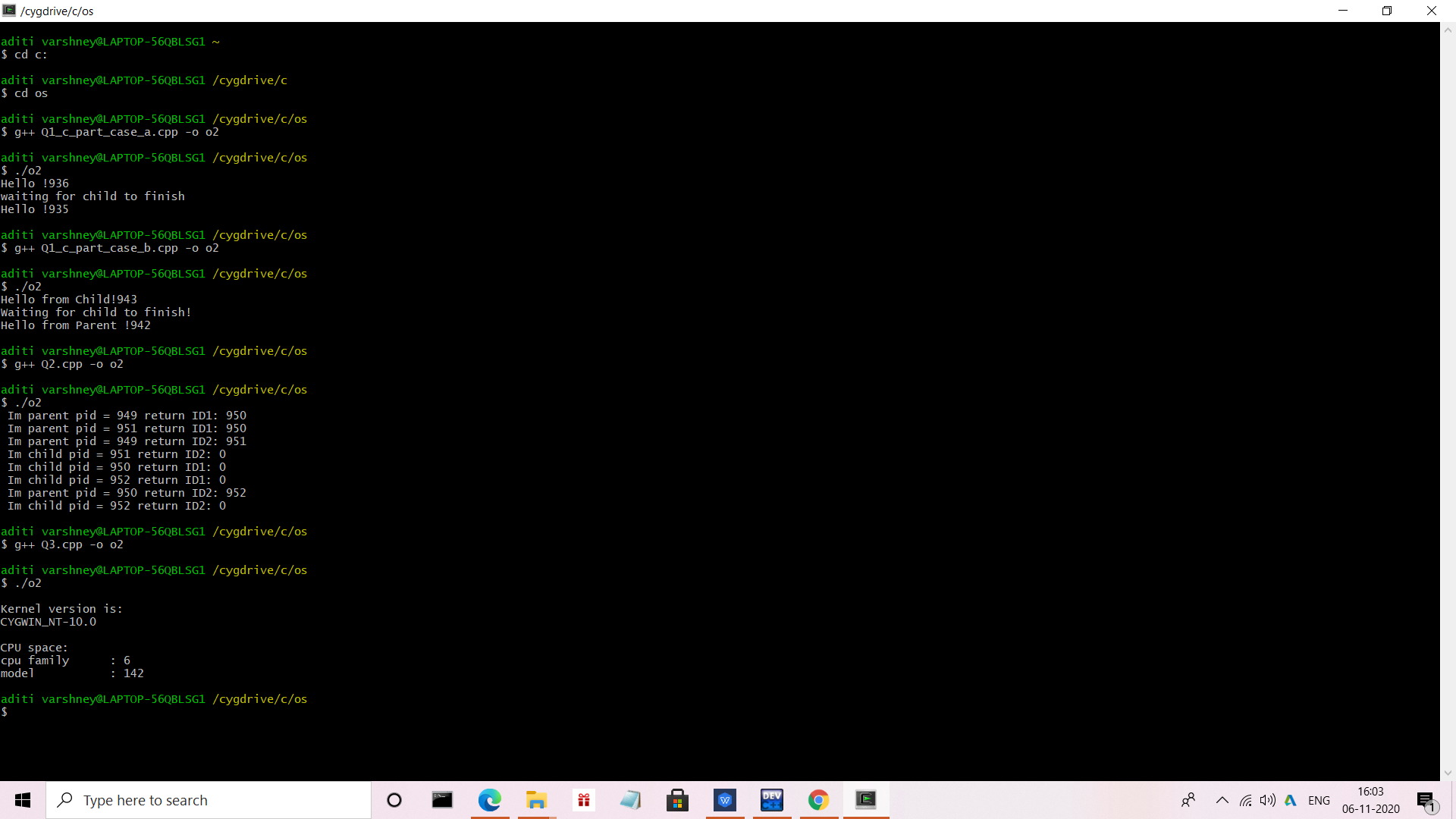
system("uname -s");

cout<<"\nCPU space: \n";

system("cat /proc/cpuinfo |awk 'NR==3,NR==4{print}' \n");

return 0;

}



4. Write a program to report behaviour of Linux kernel including information on

configured memory, amount of free and used memory. (Memory information)

Answer:-

#include<iostream>

using namespace std;

int main()

{

cout<<"\n Configured memory is :\n";

system("cat /proc/meminfo |awk 'NR==1{print $2}'\n");

cout<<"\n Amount of free memory is :\n";

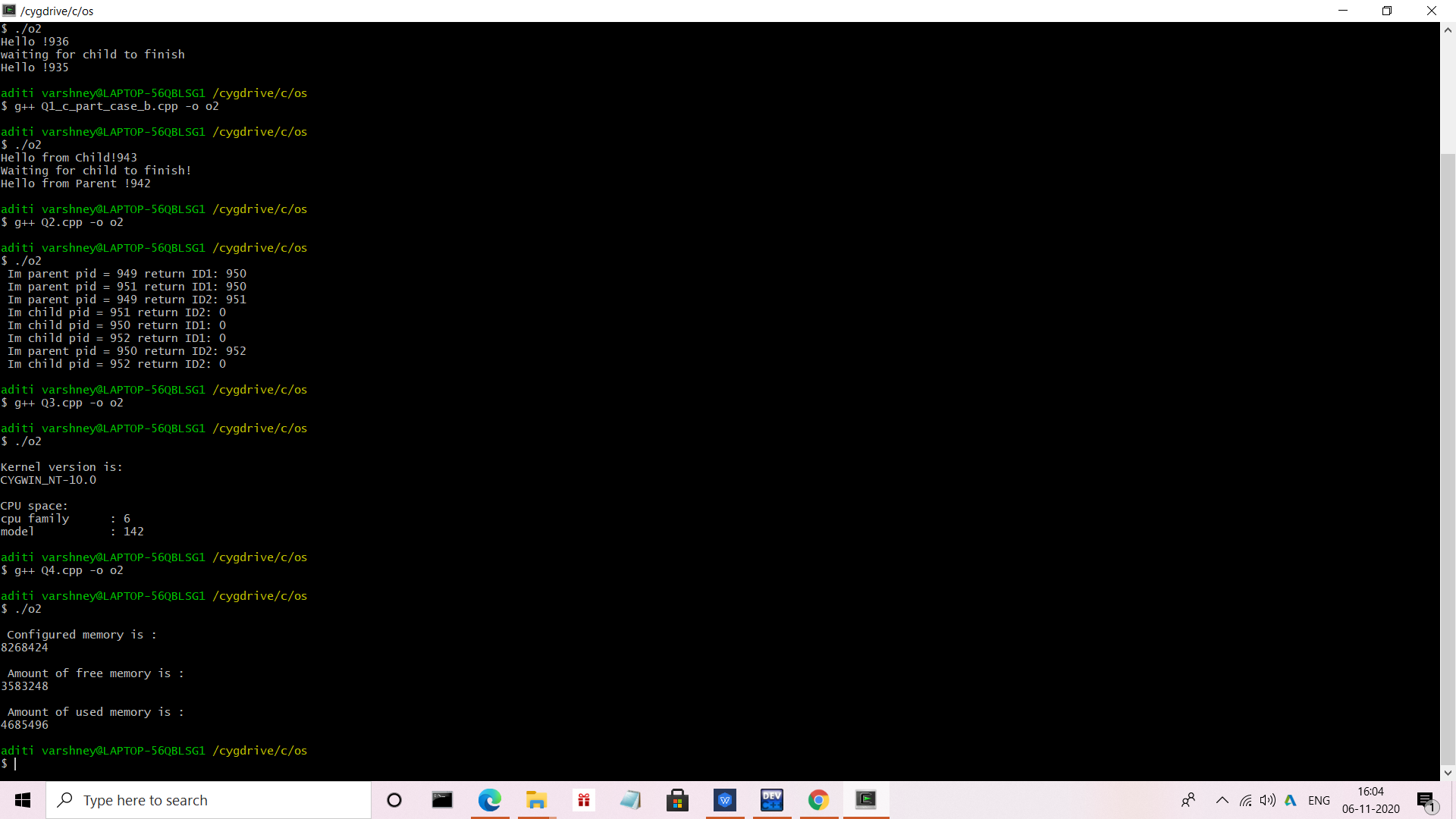
system("cat /proc/meminfo |awk 'NR==2{print $2}'\n");

cout<<"\n Amount of used memory is :\n";

system("cat /proc/meminfo |awk '{if (NR==1) a=$2; if (NR==2) b=$2 } END {print a-b}'\n");

return 0;

}



5. Write a program to print file details including owner access permissions, file access

time, where file name is given as command line argument.

Answer:-

#include<iostream>

#include<stdlib.h>

#include<stdio.h>

#include<unistd.h>

#include <sys/stat.h>

#include <sys/types.h>

using namespace std;

int main(int argc, char\*\* argv)

{

if(argc !=2)

{

cout<<"\nEnter file name!\n";

return 1;

}

struct stat fileStat;

if(stat(argv[1],&fileStat) < 0)

return 1;

cout<<"\nFile details for "<< argv[1]<<" are :\n";

cout<<"File Size: "<<fileStat.st\_size<<" bytes\n";

cout<<" time of last access is : "<<ctime(&fileStat.st\_atime);

cout<<" time of last modification is : " << ctime(&fileStat.st\_mtime);

cout<<" time of last change is : "<< ctime(&fileStat.st\_ctime);

cout<<"File Permissions: \t";

cout<<( (S\_ISDIR(fileStat.st\_mode)) ? "d" : "-");

cout<<( (fileStat.st\_mode & S\_IRUSR) ? "r" : "-");

cout<<( (fileStat.st\_mode & S\_IWUSR) ? "w" : "-");

cout<<( (fileStat.st\_mode & S\_IXUSR) ? "x" : "-");

cout<<( (fileStat.st\_mode & S\_IRGRP) ? "r" : "-");

cout<<( (fileStat.st\_mode & S\_IWGRP) ? "w" : "-");

cout<<( (fileStat.st\_mode & S\_IXGRP) ? "x" : "-");

cout<<( (fileStat.st\_mode & S\_IROTH) ? "r" : "-");

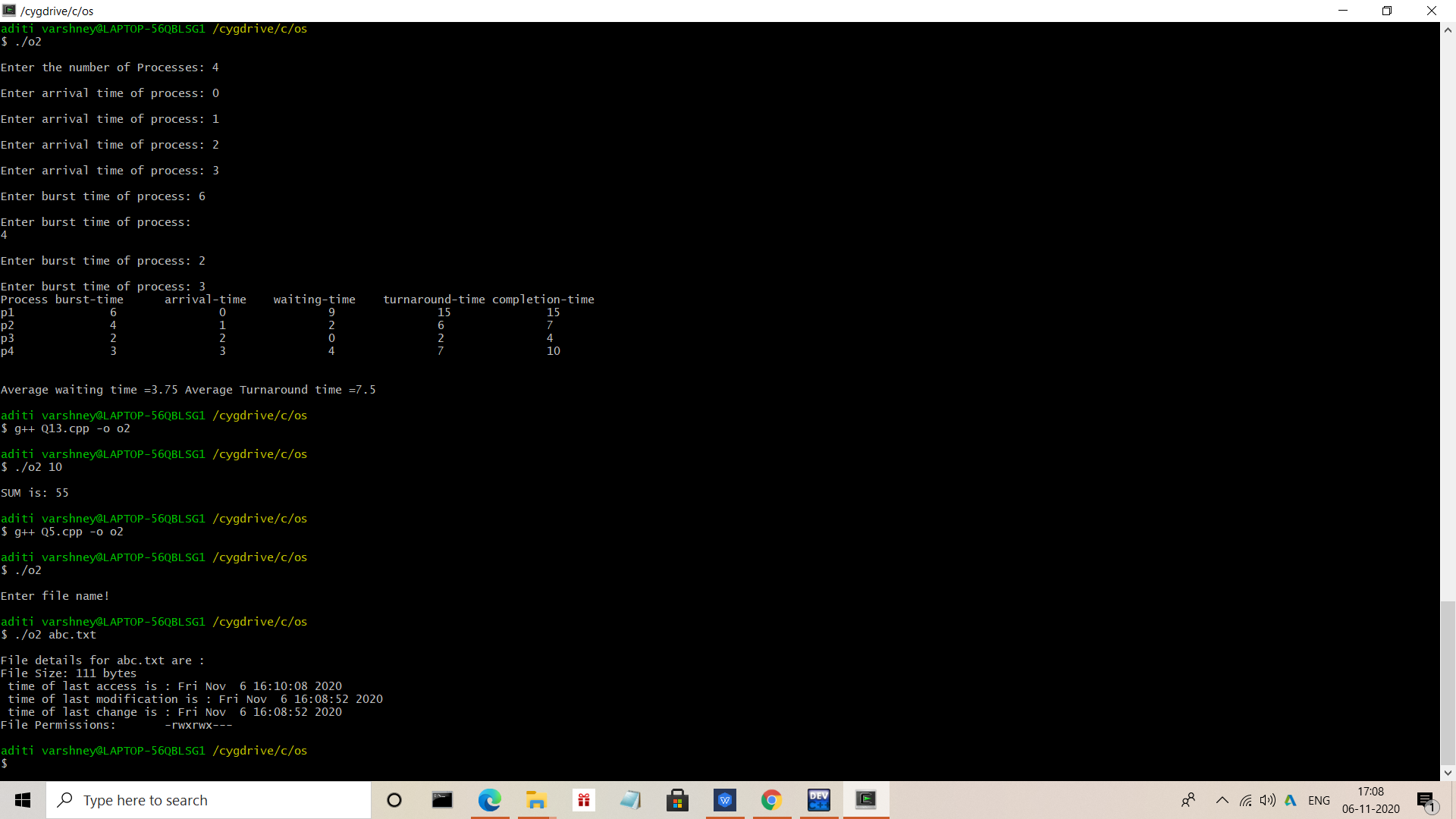
cout<<( (fileStat.st\_mode & S\_IWOTH) ? "w" : "-");

cout<<( (fileStat.st\_mode & S\_IXOTH) ? "x" : "-");

cout<<endl;

return 0;

}



6. Write a program to copy files using system calls.

Answer:-

#include <iostream>

#include <stdlib.h>

#include <fcntl.h>

#include <errno.h>

#include<unistd.h>

#include<sys/types.h>

#define BUFF\_SIZE 1024

using namespace std;

int main(int argc, char\* argv[])

{

int srcFD,destFD,nbread,nbwrite;

char \*buff[BUFF\_SIZE];

/\*Check if both src & dest files are received or --help is received to get usage\*/

if(argc != 3 || argv[1] == "--help")

{

cout<<"\nUsage: cpcmd source\_file destination\_file\n";

exit(EXIT\_FAILURE);

}

/\*Open source file\*/

srcFD = open(argv[1],O\_RDONLY);

if(srcFD == -1)

{

cout<<"\nError opening file "<<argv[1]<<" errno = \n"<<errno;

exit(EXIT\_FAILURE);

}

/\*Open destination file with respective flags & modes

O\_CREAT & O\_TRUNC is to truncate existing file or create a new file

S\_IXXXX are file permissions for the user,groups & others\*/

destFD = open(argv[2],O\_WRONLY | O\_CREAT | O\_TRUNC, S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IWGRP | S\_IROTH | S\_IWOTH);

if(destFD == -1)

{

cout<<"\nError opening file "<<argv[2]<<" errno = \n"<<errno;

exit(EXIT\_FAILURE);

}

/\*Start data transfer from src file to dest file till it reaches EOF\*/

while((nbread = read(srcFD,buff,BUFF\_SIZE)) > 0)

{

if(write(destFD,buff,nbread) != nbread)

cout<<"\nError in writing data to \n"<<argv[2];

}

if(nbread == -1)

cout<<"\nError in reading data from \n"<<argv[1];

if(close(srcFD) == -1)

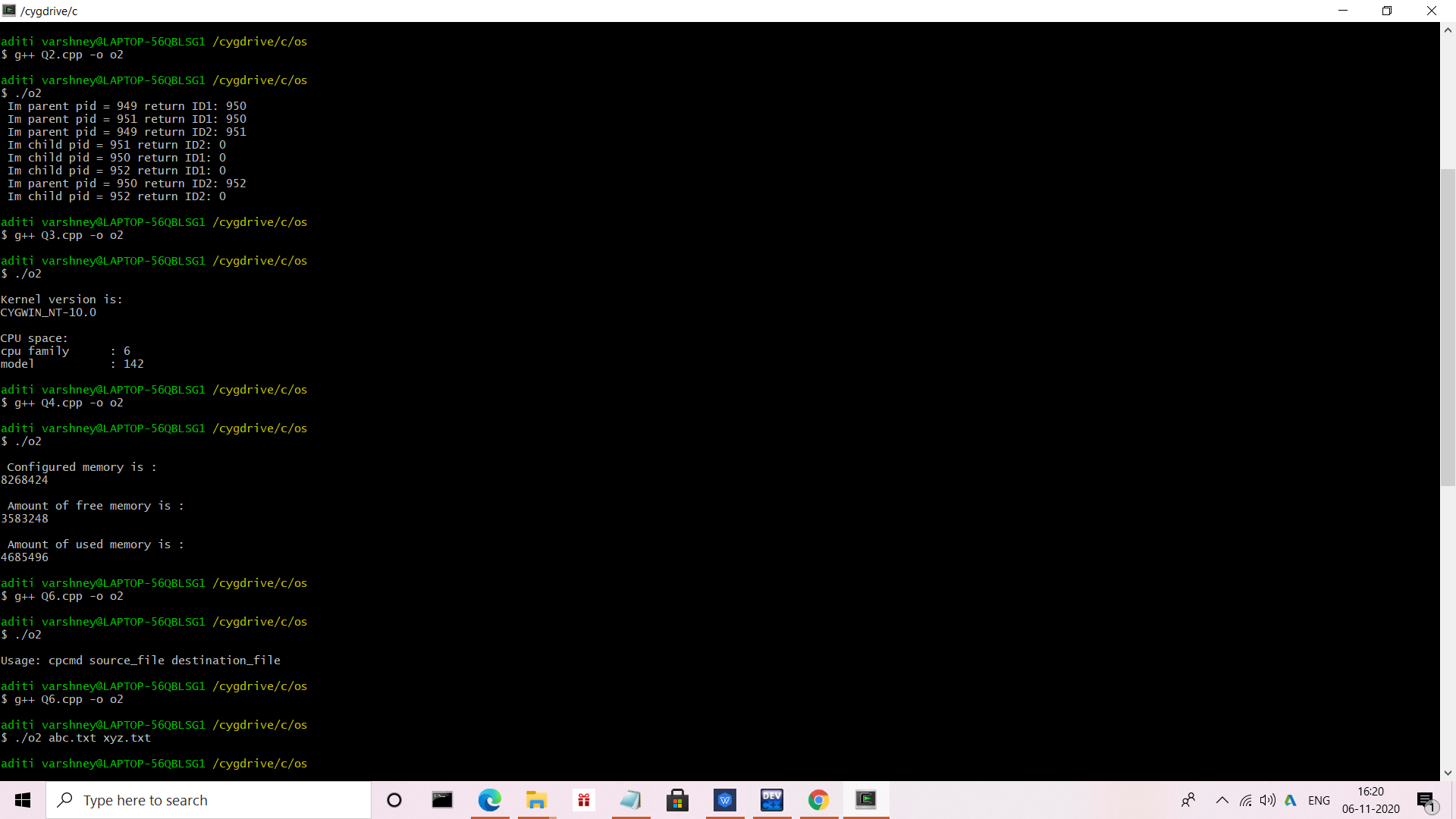
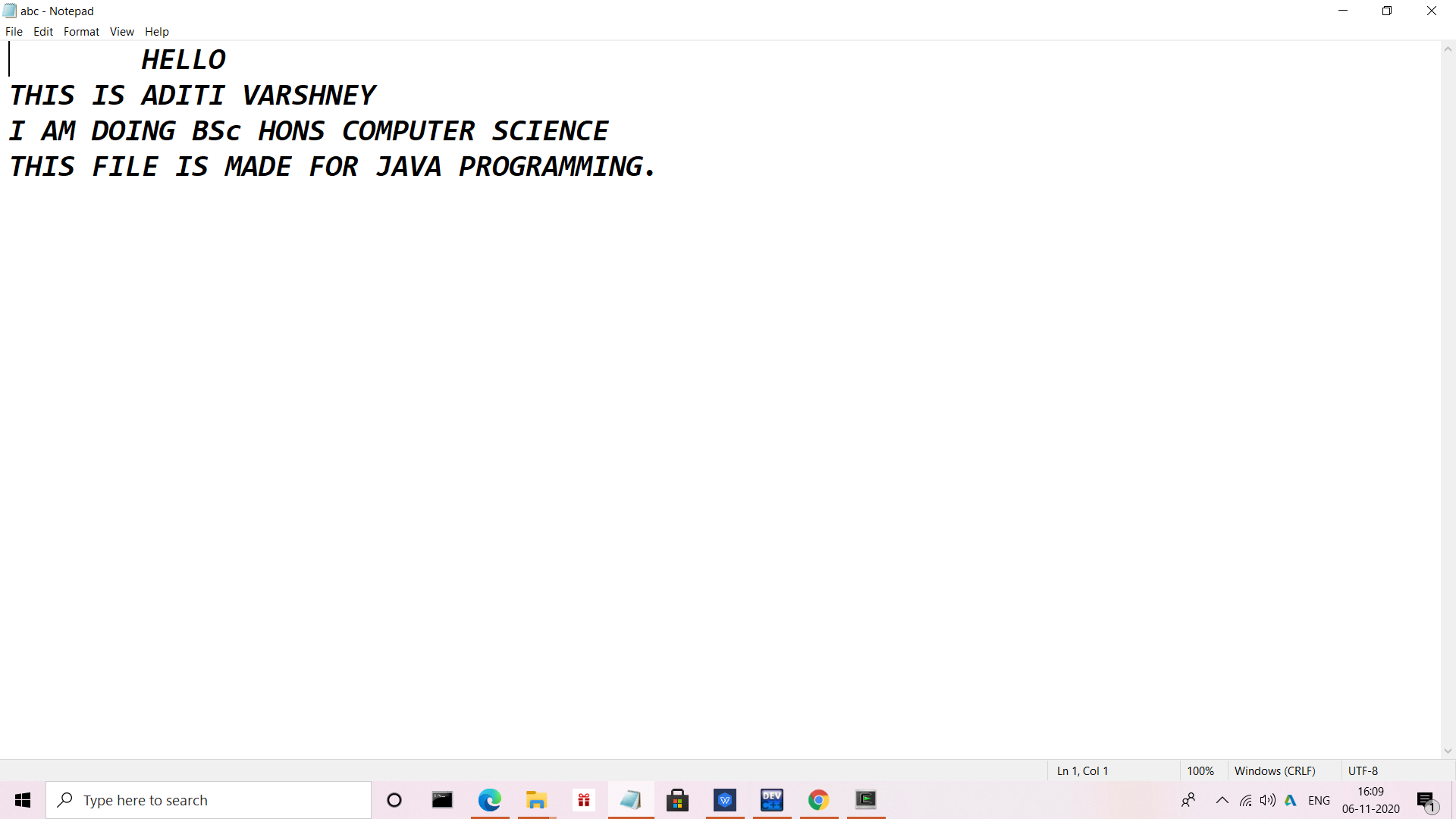
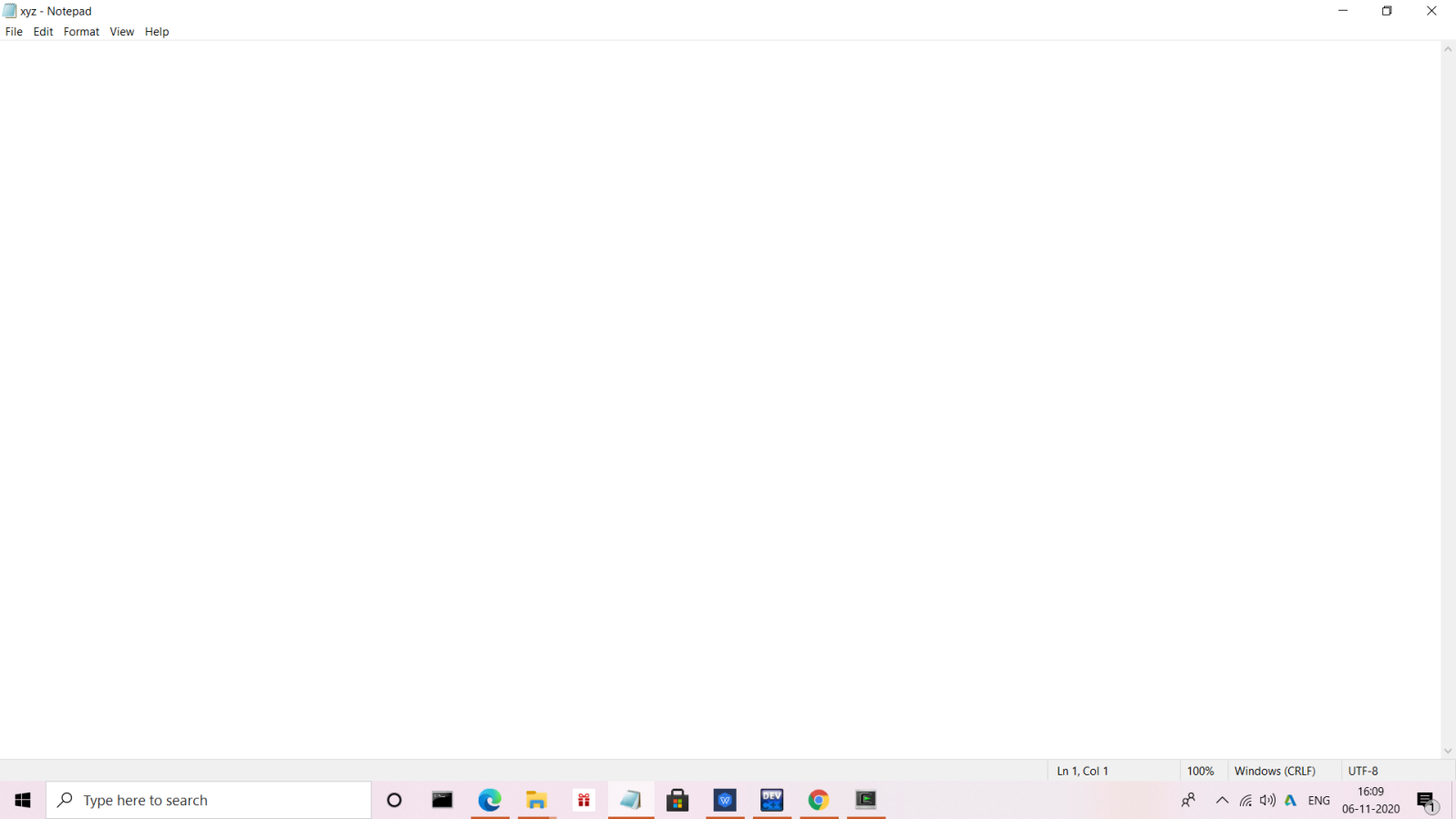
cout<<"\nError in closing file \n"<<argv[1];

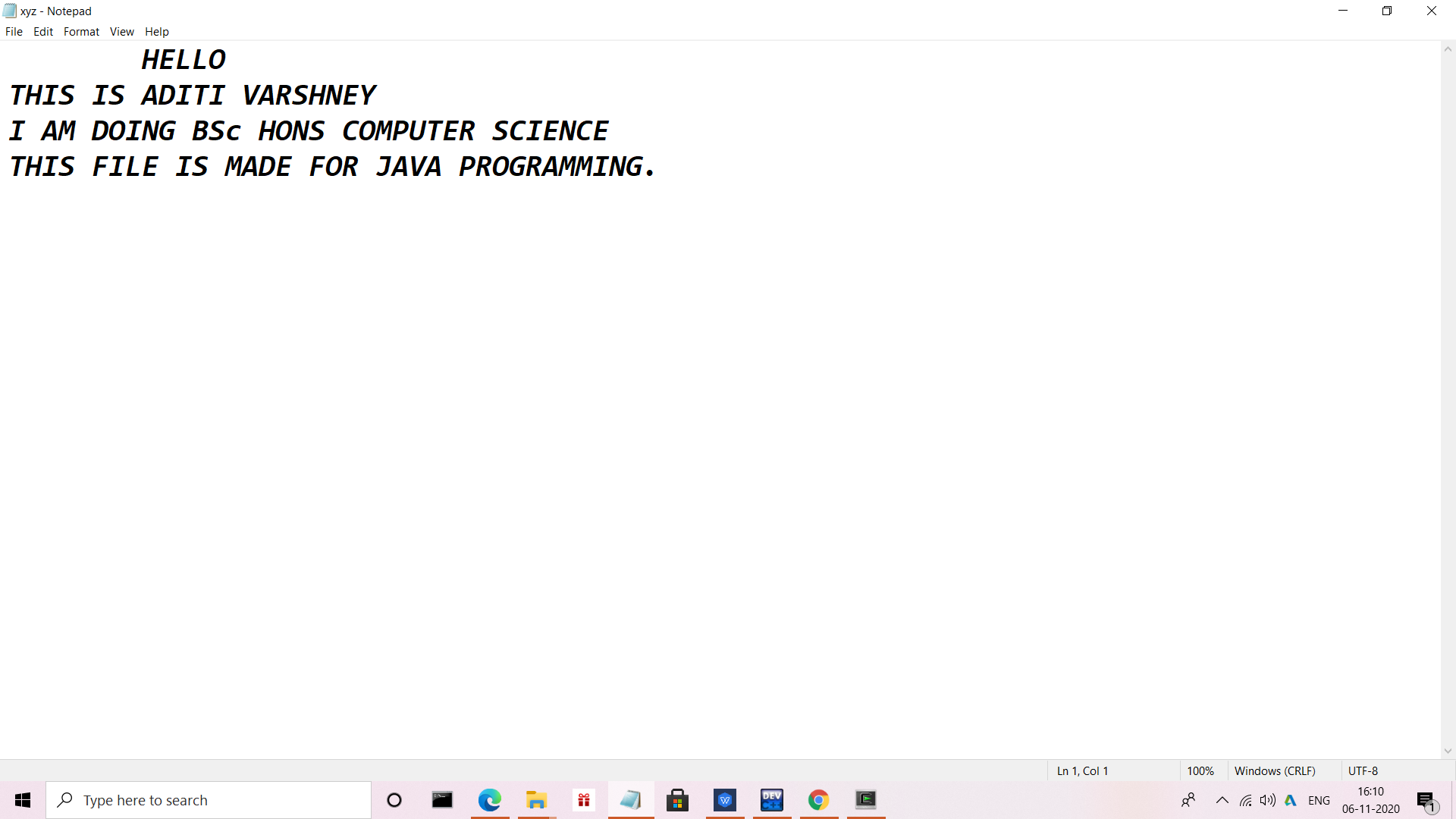
if(close(destFD) == -1)

cout<<"\nError in closing file \n"<<argv[2];

exit(EXIT\_SUCCESS);

}





7. Write a program to implement FCFS scheduling algorithm.

Answer:-

#include<iostream>

using namespace std;

class sjf

{

private:

int at[100],bt[100],tat[100],st[100],ct[100],wt[100];

int p[100];

int n;

public:

void input();

void sort();

void sort\_disp();

void clac();

void disp();

void avg();

};

void sjf::input()

{

cout<<"\n Enter number of processes :";

cin>>n;

cout<<"\n Enter the Arrival Time : \n";

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

{

cout<<"\n Process of p"<<i<<" : ";

cin>>p[i];

cout<<"\n Arrival time of p"<<i<<" : ";

cin>>at[i];

}

cout<<"\n Enter the Burst Time : \n";

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

{

cout<<"\n Burst time of p"<<i<<" : ";

cin>>bt[i];

}

}

void sjf::sort()

{

int temp,temp1;

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;

temp1=bt[i];

bt[i]=bt[j];

bt[j]=temp1;

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

}

void sjf::sort\_disp()

{

cout<<"\n \n Sorted Process : \n";

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

{

cout<<"at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<"\n";

}

}

void sjf::clac()

{

int x,t=0;

int temp;

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

{

if(t>=at[i])

{

st[i]=t;

t+=bt[i];

ct[i]=t;

i++;

}

else if(at[i+1]>t)

{

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

{

if(t>=at[k])

{

int a1;

int b1;

int c1;

int a=at[i];

int b=bt[i];

int c=p[i];

a1=at[i+1];

b1=bt[i+1];

c1=p[i+1];

at[i]=at[k];

bt[i]=bt[k];

p[i]=p[k];

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

{

if(z==(i+1))

{

at[z]=a;

bt[z]=b;

p[z]=c;

}

else

{

at[z]=a1;

bt[z]=b1;

p[z]=c1;

a1=at[z+1];

b1=bt[z+1];

c1=p[z+1];

}

}

break;

}

}

x=i;

i=x;

}

else{

temp=at[i];

at[i]=at[i+1];

at[i+1]=temp;

temp=bt[i];

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

bt[i+1]=temp;

temp=p[i];

p[i]=p[i+1];

p[i+1]=temp;

x=i;

i=x;

}

}

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

{

wt[i]=st[i]-at[i];

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

}

}

void sjf::disp()

{

cout<<"\n \n----------Scheduling Table----------- \n";

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

{

cout<<"P"<<p[i]<<"\t at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<" \t st["<<i<<"] :"<<st[i]<<" \t ct["<<i<<"] :"<<ct[i]<<" \t wt["<<i<<"] :"<<wt[i]<<" \t tat["<<i<<"] :"<<tat[i]<<"\n";

}

}

void sjf::avg()

{

float sum=0.0;

float sum1=0.0;

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

{

sum+=tat[i];

sum1+=wt[i];

}

cout<<"Average turnaround time is : "<<sum/n<<"\n";

cout<<"Average waiting time is : "<<sum1/n<<"\n";

}

int main()

{

sjf s;

s.input();

s.sort();

s.sort\_disp();

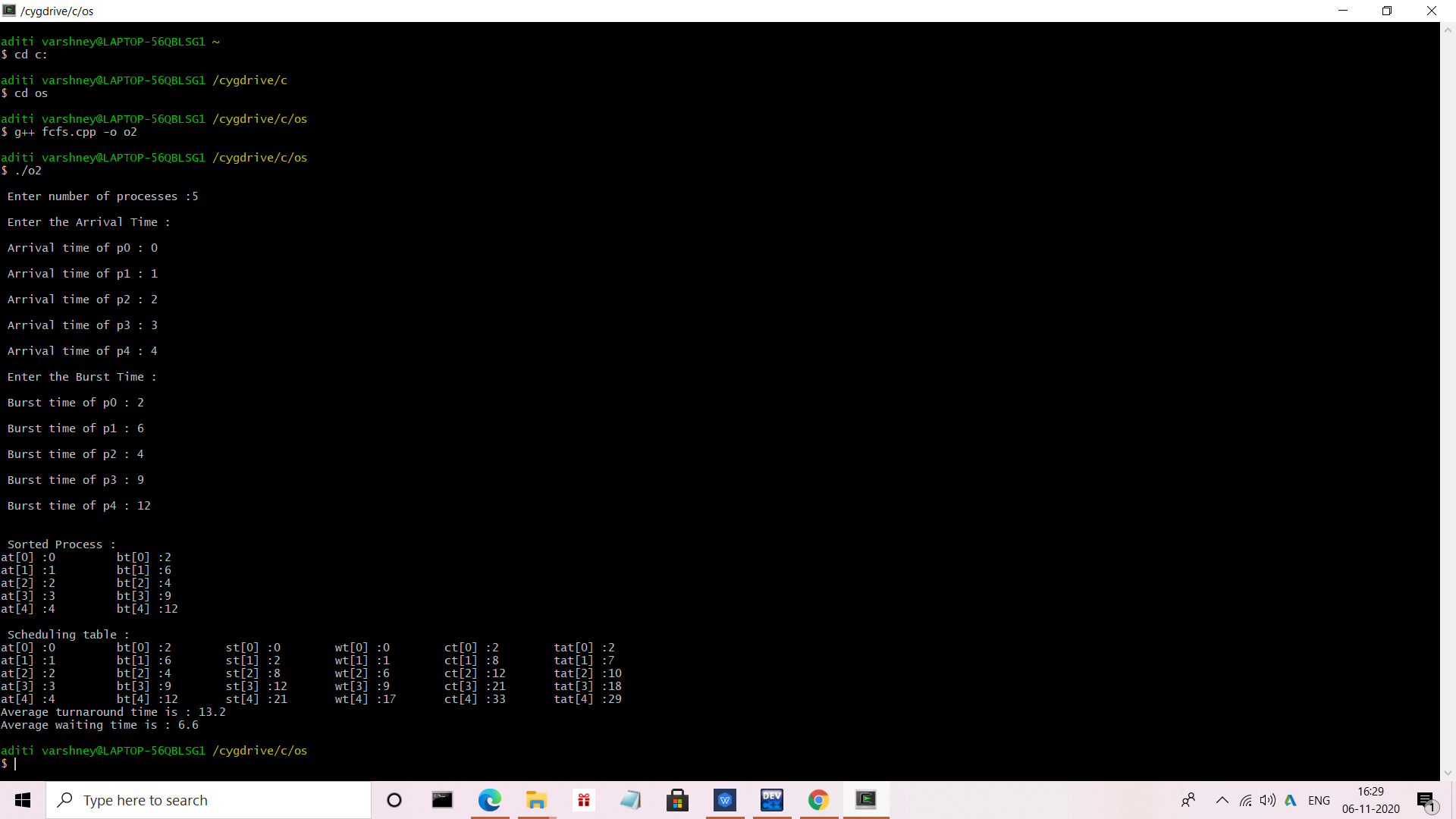
s.clac();

s.disp();

s.avg();

return 0;

}



8. Write a program to implement Round Robin scheduling algorithm.

Answer:-

#include<stdio.h>

#include<iostream>

#include<stdlib.h>

using namespace std;

int main()

{

int count,j,n,time,remain,flag=0,time\_quantum, i=0,k;

int wait\_time=0,turnaround\_time=0,at[20],bt[20],rt[20], gantt[20][2];

// Get the user enter number of processes.

cout<<"Enter Total Process:\t ";

cin>>n;

remain=n;

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

{

cout<<"Enter Arrival Time and Burst Time for Process :",count+1;

cin>>at[count];

cin>>bt[count];

rt[count]=bt[count];

}

cout<<"Enter Time Quantum:\t";

cin>>time\_quantum;

cout<<"\n\nProcess\t|Turnaround Time|Waiting Time\n\n";

//time counter is used to store the current time at which a context switch occurs

//count is initialised to be the first process and stores the current process number which is being currently operated upon

//remain is initialised to be the total number of rocesses in the queue and stores the remaining number of processes in the queue. As soon as a process

//has been executed for its entire burst time, remain gets decremented

for(time=0,count=0;remain!=0;)

{

if(rt[count]<=time\_quantum && rt[count]>0) // if remaining time of a process is smaller than time quantum, then increment time by remaining time of the process

{

time+=rt[count];

rt[count]=0;

gantt[i][0]= count;

gantt[i][1]= time;

i++;

flag=1;

}

else if(rt[count]>0) // if remaining time of the process is larger than the time quantum

{

rt[count]-=time\_quantum;

time+=time\_quantum;

gantt[i][0]= count;

gantt[i][1]= time;

i++;

}

if(rt[count]==0 && flag==1) // if remaining time of the process becomes zero after getting turn in this iteration

{

remain--;

cout<<"P["<<count+1<<"]\t|\t"<<time-at[count]<<"\t|\t"<<time-at[count]-bt[count]<<"\n";

wait\_time+=time-at[count]-bt[count];

turnaround\_time+=time-at[count];

flag=0;

}

if(count==n-1)

count=0;

else if(at[count+1]<=time)// whether the next process in queue has arrived on or before the current time, ie , is there any process in the queue

count++; //go to that process and repeat above steps

else//remain on the same process

count=0;

}

cout<<"\nAverage Waiting Time= "<<wait\_time\*1.0/n<<endl;

cout<<"Avg Turnaround Time = "<<turnaround\_time\*1.0/n<<endl;

cout<<endl<<"---------Gantt Chart---------"<<endl<<"|PID\t|End Time\t|"<<endl;

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

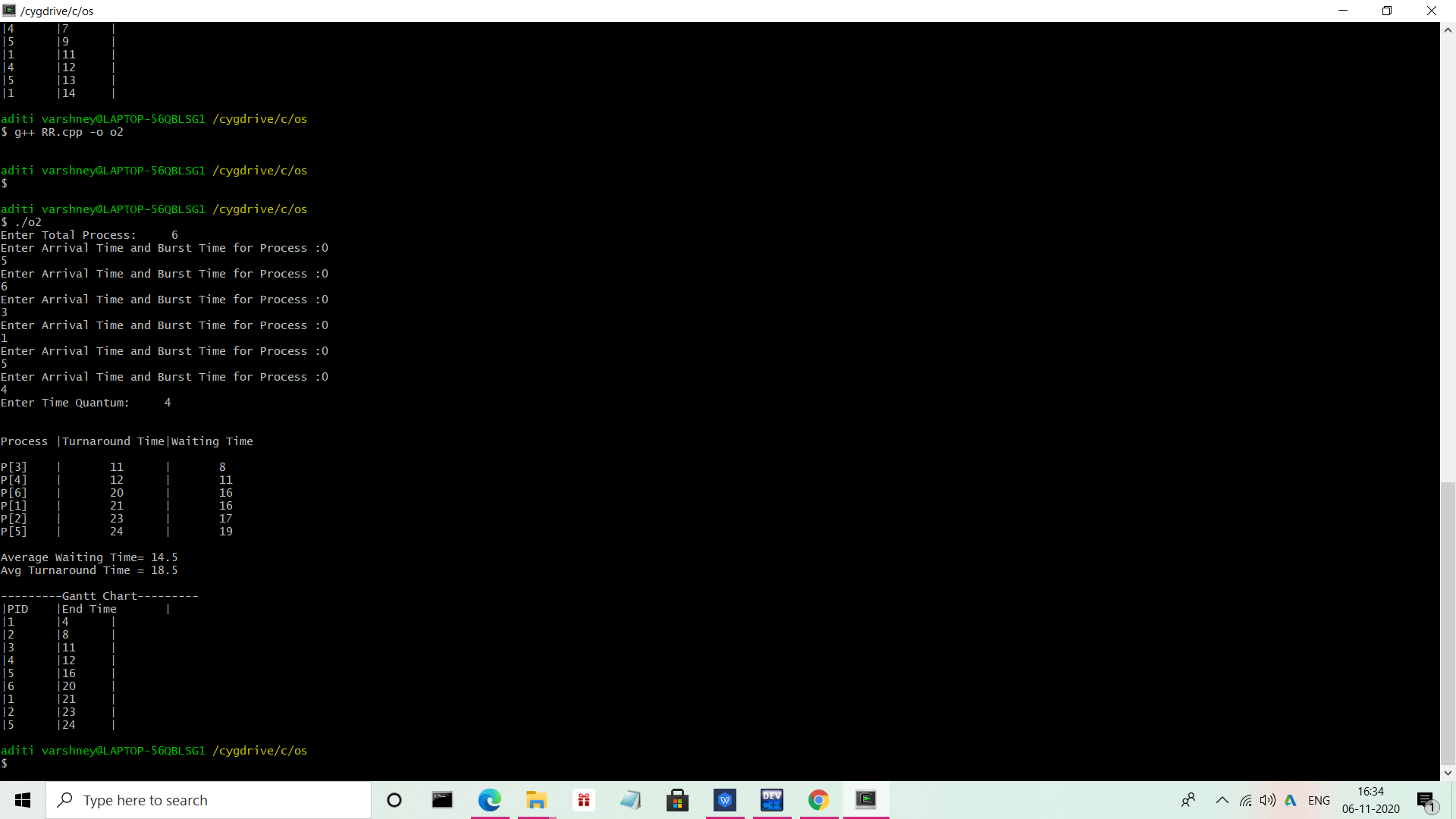
{

cout<<"|"<<gantt[k][0]+1<<"\t|"<<gantt[k][1]<<"\t|"<<endl;

}

return 0;

}



9. Write a program to implement SJF scheduling algorithm.

Answer:-

#include<iostream>

using namespace std;

class sjf

{

private:

int at[100],bt[100],tat[100],st[100],ct[100],wt[100];

int p[100];

int n;

public:

void input();

void sort();

void sort\_disp();

void clac();

void disp();

void avg();

};

void sjf::input()

{

cout<<"\n Enter number of processes :";

cin>>n;

cout<<"\n Enter the Arrival Time : \n";

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

{

cout<<"\n Process of p"<<i<<" : ";

cin>>p[i];

cout<<"\n Arrival time of p"<<i<<" : ";

cin>>at[i];

}

cout<<"\n Enter the Burst Time : \n";

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

{

cout<<"\n Burst time of p"<<i<<" : ";

cin>>bt[i];

}

}

void sjf::sort()

{

int temp,temp1;

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;

temp1=bt[i];

bt[i]=bt[j];

bt[j]=temp1;

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

}

void sjf::sort\_disp()

{

cout<<"\n \n Sorted Process : \n";

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

{

cout<<"at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<"\n";

}

}

void sjf::clac()

{

int x,t=0;

int temp;

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

{

if(t>=at[i])

{

st[i]=t;

t+=bt[i];

ct[i]=t;

i++;

}

else if(at[i+1]>t)

{

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

{

if(t>=at[k])

{

int a1;

int b1;

int c1;

int a=at[i];

int b=bt[i];

int c=p[i];

a1=at[i+1];

b1=bt[i+1];

c1=p[i+1];

at[i]=at[k];

bt[i]=bt[k];

p[i]=p[k];

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

{

if(z==(i+1))

{

at[z]=a;

bt[z]=b;

p[z]=c;

}

else

{

at[z]=a1;

bt[z]=b1;

p[z]=c1;

a1=at[z+1];

b1=bt[z+1];

c1=p[z+1];

}

}

break;

}

}

x=i;

i=x;

}

else{

temp=at[i];

at[i]=at[i+1];

at[i+1]=temp;

temp=bt[i];

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

bt[i+1]=temp;

temp=p[i];

p[i]=p[i+1];

p[i+1]=temp;

x=i;

i=x;

}

}

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

{

wt[i]=st[i]-at[i];

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

}

}

void sjf::disp()

{

cout<<"\n \n----------Scheduling Table----------- \n";

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

{

cout<<"P"<<p[i]<<"\t at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<" \t st["<<i<<"] :"<<st[i]<<" \t ct["<<i<<"] :"<<ct[i]<<" \t wt["<<i<<"] :"<<wt[i]<<" \t tat["<<i<<"] :"<<tat[i]<<"\n";

}

}

void sjf::avg()

{

float sum=0.0;

float sum1=0.0;

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

{

sum+=tat[i];

sum1+=wt[i];

}

cout<<"Average turnaround time is : "<<sum/n<<"\n";

cout<<"Average waiting time is : "<<sum1/n<<"\n";

}

int main()

{

sjf s;

s.input();

s.sort();

s.sort\_disp();

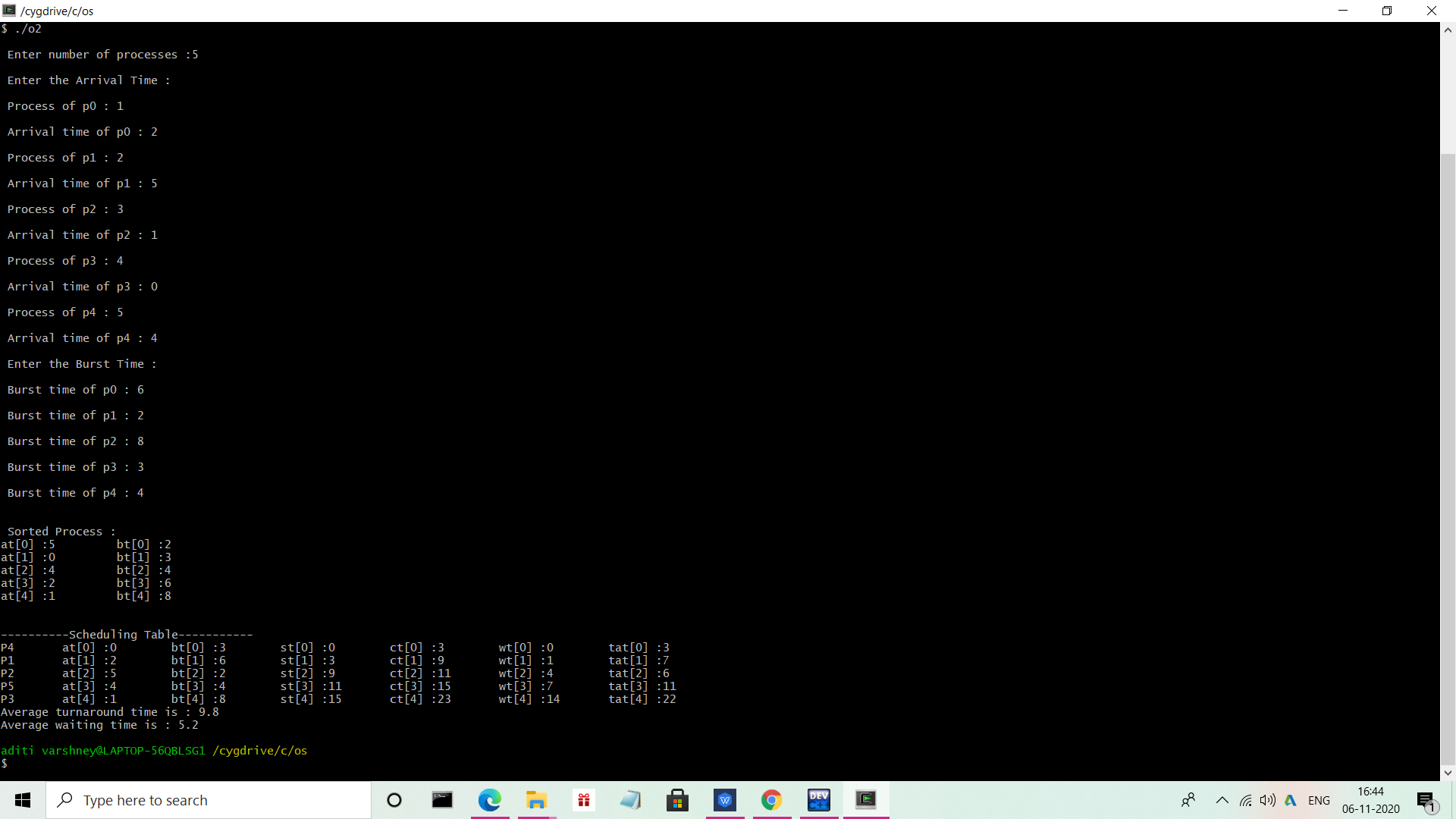
s.clac();

s.disp();

s.avg();

return 0;

}



10. Write a program to implement non-preemptive priority based scheduling algorithm.

Answer:-

#include<iostream>

using namespace std;

class prio

{

private:

int at[100],bt[100],pri[100],tat[100],st[100],ct[100],wt[100];

int p[100];

int n;

public:

void input();

void sort();

void sort\_disp();

void clac();

void disp();

void avg();

};

void prio::input()

{

cout<<"\n Enter number of processes :";

cin>>n;

cout<<"\n Enter the Arrival Time : \n";

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

{

cout<<"\n Process of p"<<i<<" : ";

cin>>p[i];

cout<<"\n Arrival time of p"<<i<<" : ";

cin>>at[i];

}

cout<<"\n Enter the Burst Time : \n";

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

{

cout<<"\n Burst time of p"<<i<<" : ";

cin>>bt[i];

}

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

{

cout<<"\n Priority of p"<<i<<" : ";

cin>>pri[i];

}

}

void prio::sort()

{

int temp,temp1;

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

{

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

{

if(pri[i]>pri[j])

{

temp=pri[i];

pri[i]=pri[j];

pri[j]=temp;

temp=at[i];

at[i]=at[j];

at[j]=temp;

temp1=bt[i];

bt[i]=bt[j];

bt[j]=temp1;

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

}

}

void prio::sort\_disp()

{

cout<<"\n \n Sorted Process : \n";

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

{

cout<<"at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<" \t pri["<<i<<"] :"<<pri[i]<<"\n";

}

}

void prio::clac()

{

int x,t=0;

int temp;

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

{

if(t>=at[i])

{

st[i]=t;

t+=bt[i];

ct[i]=t;

i++;

}

else if(at[i+1]>t)

{

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

{

if(t>=at[k])

{

int a1;

int b1;

int c1;

int d1;

int a=at[i];

int b=bt[i];

int c=p[i];

int d=pri[i];

a1=at[i+1];

b1=bt[i+1];

c1=p[i+1];

d1=pri[i+1];

at[i]=at[k];

bt[i]=bt[k];

p[i]=p[k];

pri[i]=pri[k];

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

{

if(z==(i+1))

{

at[z]=a;

bt[z]=b;

p[z]=c;

pri[z]=d;

}

else

{

at[z]=a1;

bt[z]=b1;

p[z]=c1;

pri[z]=d;

a1=at[z+1];

b1=bt[z+1];

c1=p[z+1];

d1=pri[z+1];

}

}

break;

}

}

x=i;

i=x;

}

else{

temp=pri[i];

pri[i]=pri[i+1];

pri[i+1]=temp;

temp=at[i];

at[i]=at[i+1];

at[i+1]=temp;

temp=bt[i];

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

bt[i+1]=temp;

temp=p[i];

p[i]=p[i+1];

p[i+1]=temp;

x=i;

i=x;

}

}

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

{

wt[i]=st[i]-at[i];

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

}

}

void prio::disp()

{

cout<<"\n \n----------Scheduling Table----------- \n";

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

{

cout<<"P"<<p[i]<<"\t at["<<i<<"] :"<<at[i]<<" \t bt["<<i<<"] :"<<bt[i]<<" \t pri["<<i<<"] :"<<pri[i]<<" \t st["<<i<<"] :"<<st[i]<<" \t ct["<<i<<"] :"<<ct[i]<<" \t wt["<<i<<"] :"<<wt[i]<<" \t tat["<<i<<"] :"<<tat[i]<<"\n";

}

}

void prio::avg()

{

float sum=0.0;

float sum1=0.0;

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

{

sum+=tat[i];

sum1+=wt[i];

}

cout<<"Average turnaround time is : "<<sum/n<<"\n";

cout<<"Average waiting time is : "<<sum1/n<<"\n";

}

int main()

{

prio s;

s.input();

s.sort();

s.sort\_disp();

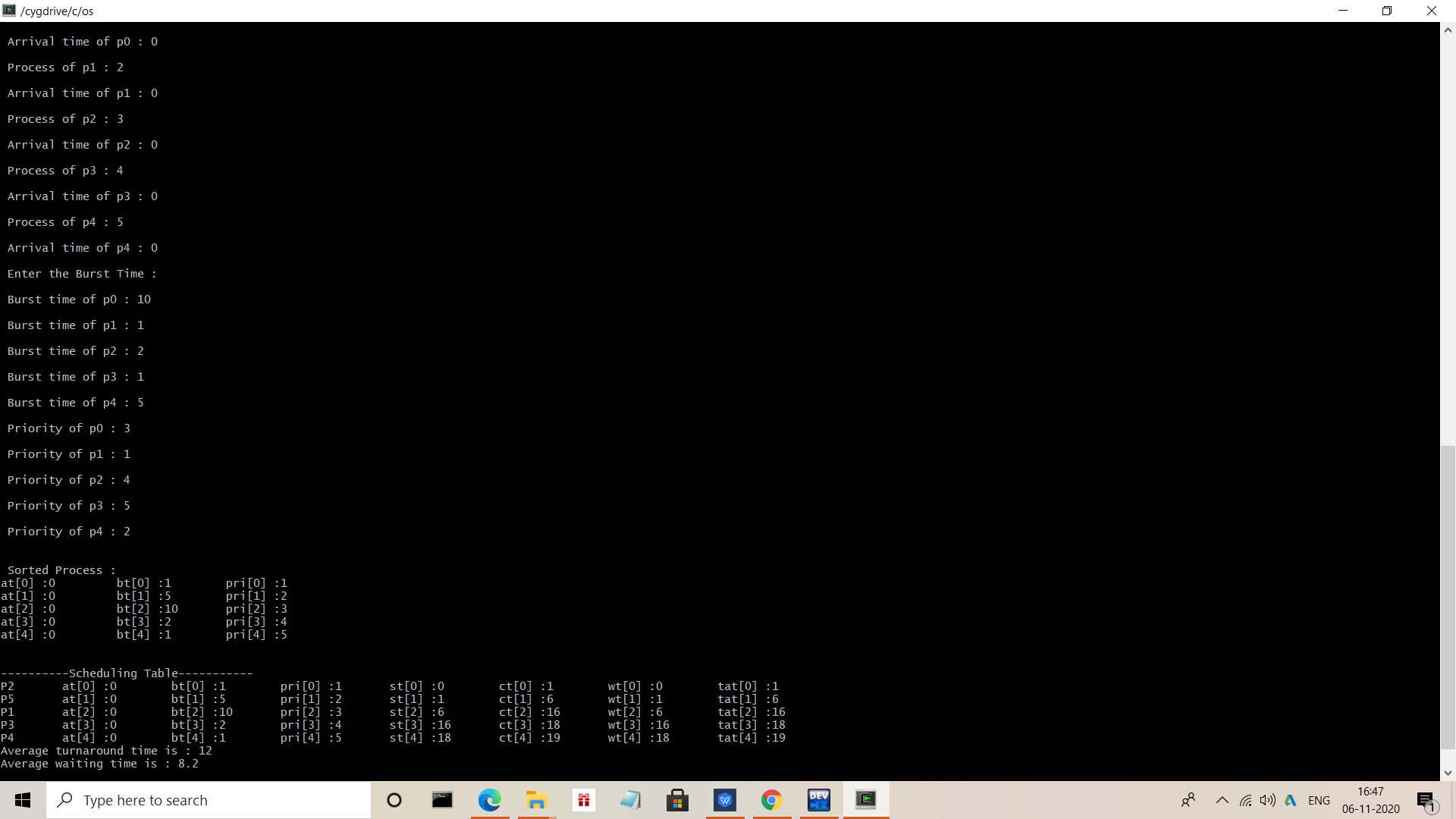
s.clac();

s.disp();

s.avg();

return 0;

}



11. Write a program to implement preemptive priority based scheduling algorithm.

Answer:-

#include<iostream>

using namespace std;

int main()

{

int a[10],b[10],x[10];

int waiting[10],turnaround[10],completion[10],p[10];

int i,j,smallest,count=0,time,n;

double avg=0,tt=0,end;

cout<<"\nEnter the number of Processes: ";

cin>>n;

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

{

cout<<"\nEnter arrival time of process: ";

cin>>a[i];

}

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

{

cout<<"\nEnter burst time of process: ";

cin>>b[i];

}

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

{

cout<<"\nEnter priority of process: ";

cin>>p[i];

}

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

x[i]=b[i];

p[9]=-1;

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

{

smallest=9;

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

{

if(a[i]<=time && p[i]>p[smallest] && b[i]>0 )

smallest=i;

}

b[smallest]--;

if(b[smallest]==0)

{

count++;

end=time+1;

completion[smallest] = end;

waiting[smallest] = end - a[smallest] - x[smallest];

turnaround[smallest] = end - a[smallest];

}

}

cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time" <<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-time"<<"\t"<<"Priority"<<endl;

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

{

cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<"\t\t"<<turnaround[i]<<"\t\t"<<completion[i]<<"\t\t"<<p[i]<<endl;

avg = avg + waiting[i];

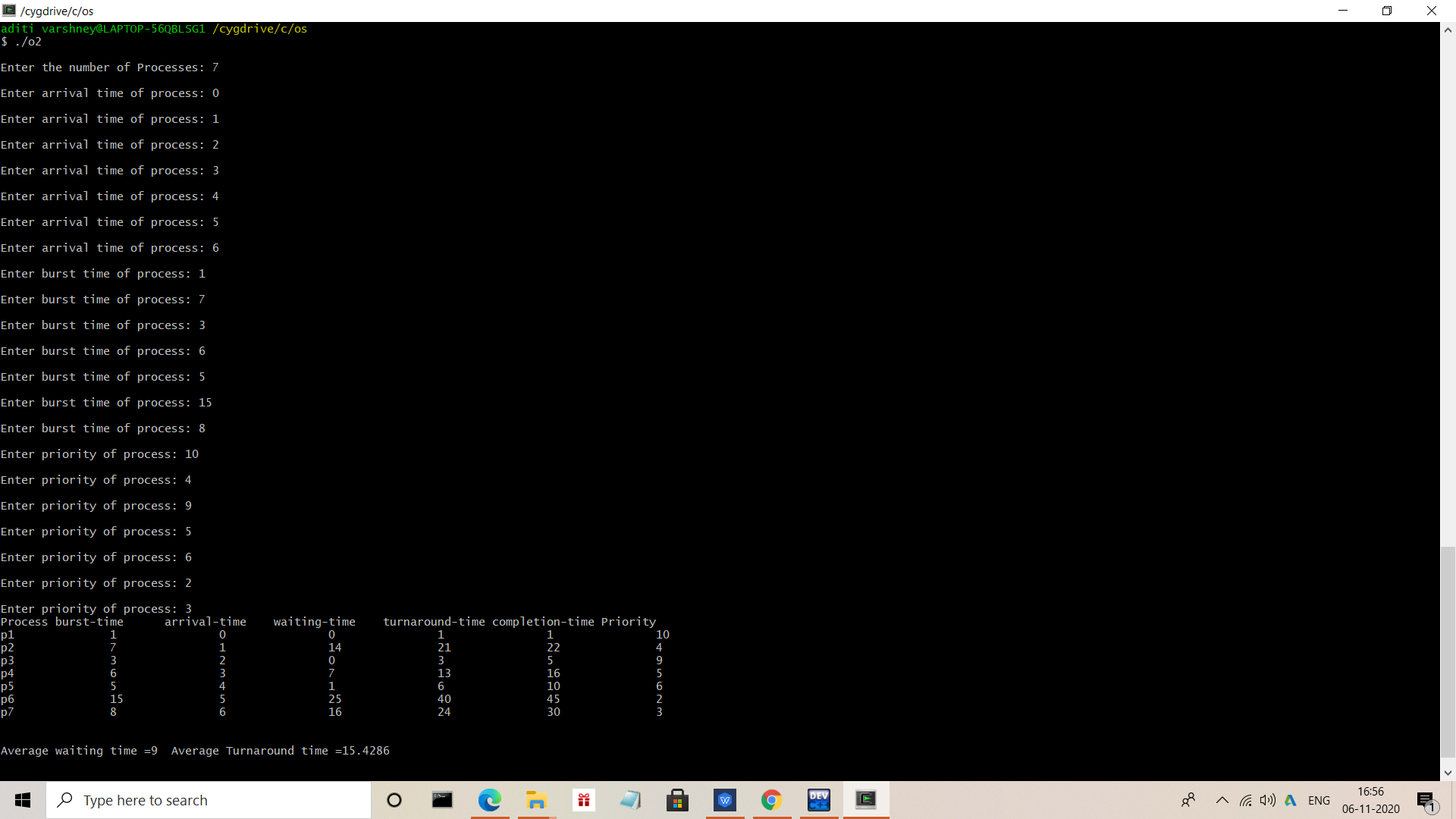
tt = tt + turnaround[i];

}

cout<<"\n\nAverage waiting time ="<<avg/n;

cout<<" Average Turnaround time ="<<tt/n<<endl;

}



12. Write a program to implement SRTF scheduling algorithm.

Answer:-

#include<iostream>

using namespace std;

int main()

{

int a[10],b[10],x[10];

int waiting[10],turnaround[10],completion[10];

int i,j,smallest,count=0,time,n;

double avg=0,tt=0,end;

cout<<"\nEnter the number of Processes: "; //input

cin>>n;

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

{

cout<<"\nEnter arrival time of process: "; //input

cin>>a[i];

}

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

{

cout<<"\nEnter burst time of process: "; //input

cin>>b[i];

}

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

x[i]=b[i];

b[9]=9999;

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

{

smallest=9;

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

{

if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )

smallest=i;

}

b[smallest]--;

if(b[smallest]==0)

{

count++;

end=time+1;

completion[smallest] = end;

waiting[smallest] = end - a[smallest] - x[smallest];

turnaround[smallest] = end - a[smallest];

}

}

cout<<"Process"<<"\t"<< "burst-time"<<"\t"<<"arrival-time"

<<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<<

"\t"<<"completion-time"<<endl;

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

{

cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<

"\t\t"<<turnaround[i]<<"\t\t"<<completion[i]<<endl;

avg = avg + waiting[i];

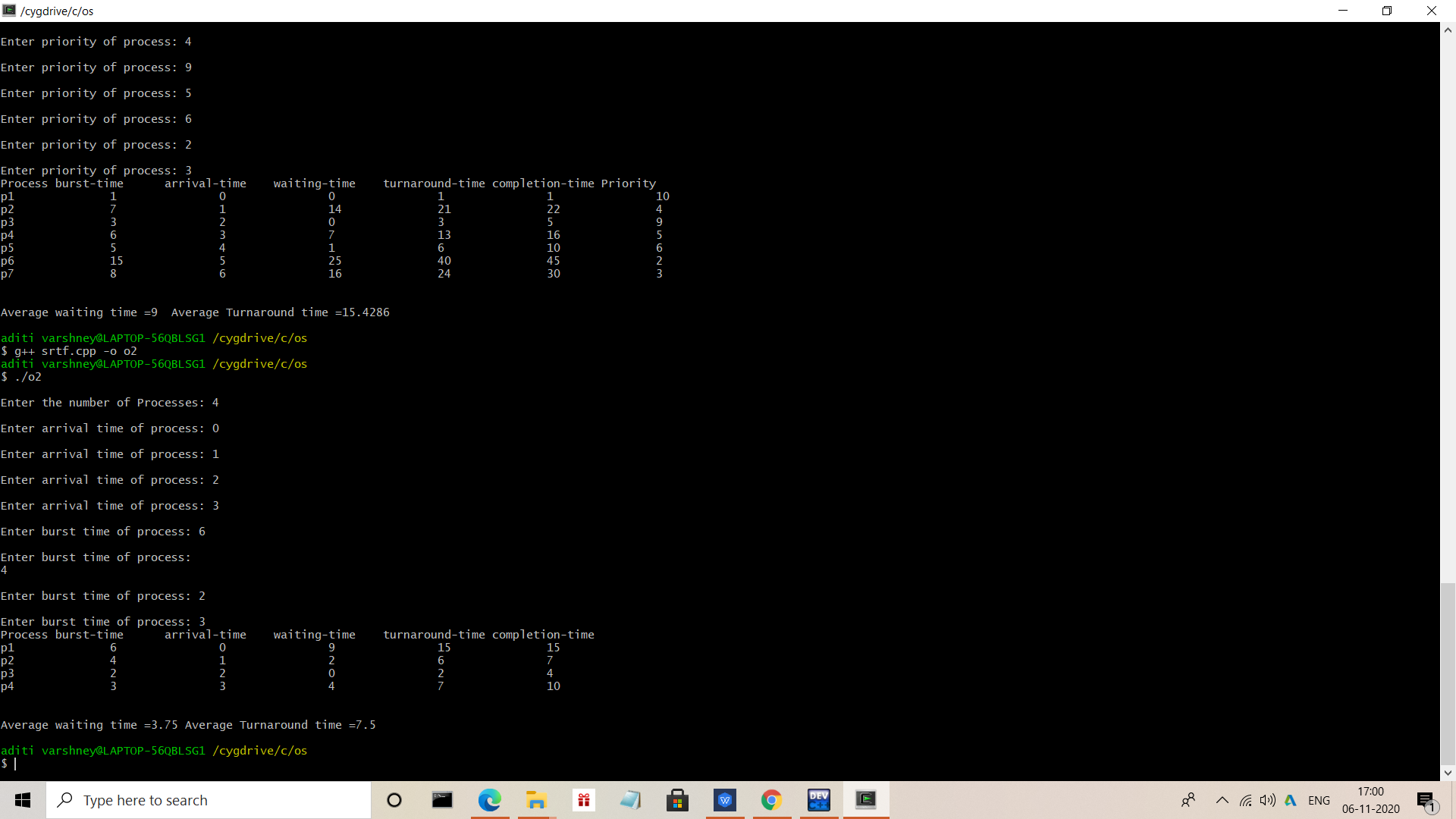
tt = tt + turnaround[i];

}

cout<<"\n\nAverage waiting time ="<<avg/n;

cout<<" Average Turnaround time ="<<tt/n<<endl;

}



13. Write a program to calculate sum of n numbers using thread library.

Answer:-

#include<pthread.h>

#include<iostream>

using namespace std;

int sum;

void\* runner(void\* param);

int main(int argc,char \*argv[])

{

pthread\_t tid;

pthread\_attr\_t attr;

if(argc!=2)

{

cout<<"\nUsage :a.out<integer value>\n";

return -1;

}

if(atoi(argv[1])<0)

{

cout<<"\n%d must be >=0\n"<<atoi((const char\*)(argv[1]))<<endl;

return -1;

}

pthread\_attr\_init(&attr); //get the default attributes

pthread\_create(&tid,&attr,runner,argv[1]);//create the thread:

pthread\_join(tid,NULL);//parent waits for the child thread to finish

cout<<"\nSUM is: "<<sum<<endl;//output the value of shared data "sum"

return 0;

}

//child thread will begin execution here:

void\* runner(void\* param)

{

int i,upper=atoi((const char\*)param);

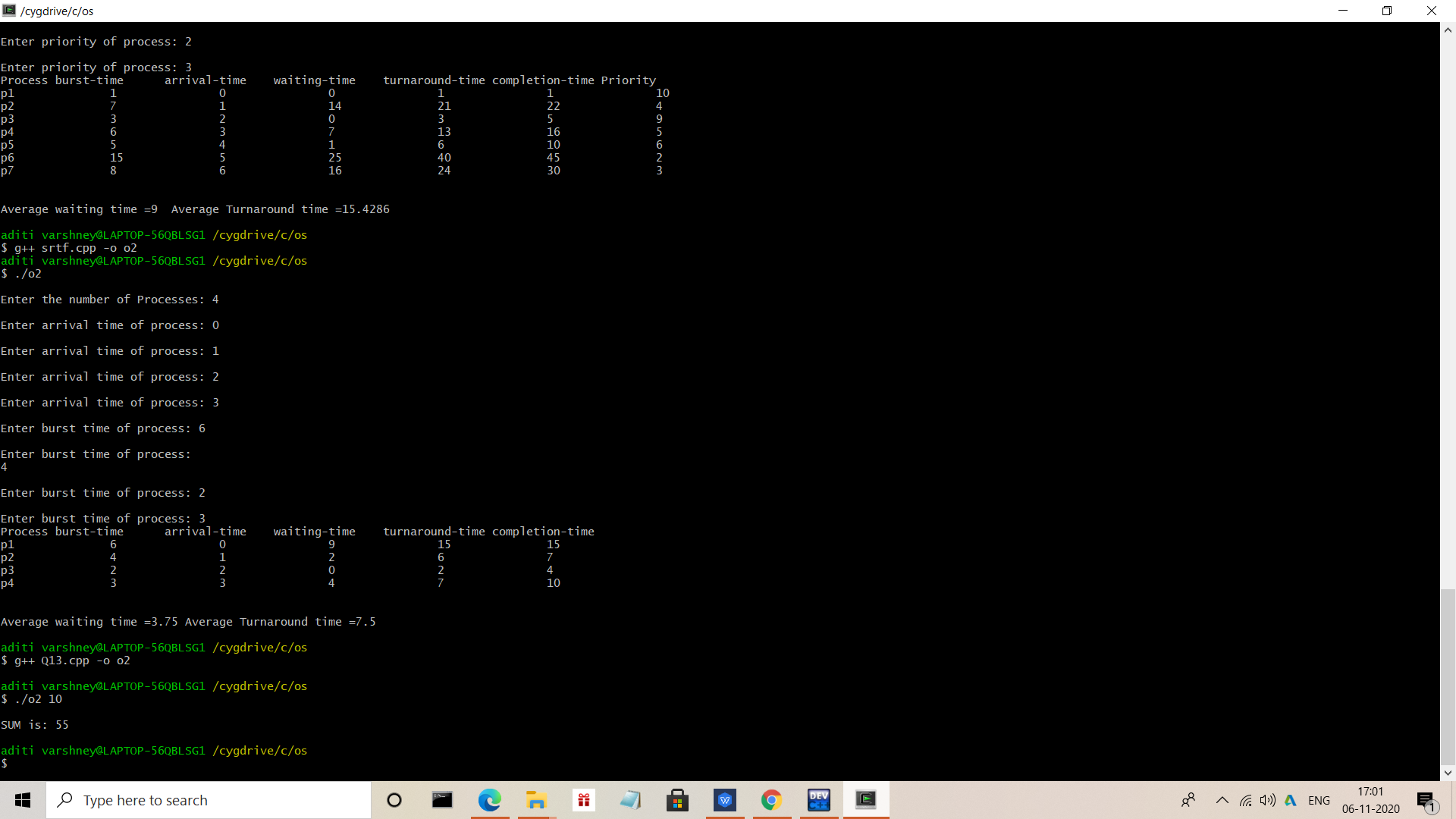
sum=0;

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

sum+=i;

pthread\_exit(0);

}



14. Write a program to implement first-fit, best-fit and worst-fit allocation strategies.

Answer:-

#include<conio.h>

#include<iostream>

using namespace std;

int main()

{

int c,i,j,k,n,l,m[10],p[10],po[20],flag,z,y,temp,temp1;

char r;

do{

cout<<"enter memory partition:\t";

cin>>n;

cout<<"\nenter memory size for\n";

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

{

cout<<"\npartition "<<i<<" :\t";

cin>>m[i];

po[i]=i;

}

cout<<"\nenter process:\t";

cin>>j;

cout<<"\nenter memory size for\n";

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

{

cout<<"\nprocess "<<i<<" :\t";

cin>>p[i];

}

cout<<"\n\*\*\*\*\*\*\*\* welcome to menu driven program of memory management\*\*\*\*\*\*\*\*\*\*\n";

cout<<"1.first fit\n";

cout<<"2.best fit\n";

cout<<"3.worst fit\n";

cout<<"enter choice:\t";

cin>>c;

switch(c)

{

case 1:

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

{

flag=1;

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

{

if(p[i]<=m[k])

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB allocated at memory partition:\t"<<po[k];

m[k]=m[k]-p[i];

break;

}

else

{

flag++;

}

}

if(flag>n)

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB can't be allocated";

}

}

break;

case 2:

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

{

for(z=y;z<=n;z++)

{

if(m[y]>m[z])

{

temp=m[y];

m[y]=m[z];

m[z]=temp;

temp1=po[y];

po[y]=po[z];

po[z]=temp1;

}

}

}

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

{

flag=1;

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

{

if(p[i]<=m[k])

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB allocated at memory partition:\t"<<po[k];

m[k]=m[k]-p[i];

break;

}

else

{

flag++;

}

}

if(flag>n)

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB can't be allocated";

}

}

break;

case 3:

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

{

for(z=y;z<=n;z++)

{

if(m[y]<m[z])

{

temp=m[y];

m[y]=m[z];

m[z]=temp;

temp1=po[y];

po[y]=po[z];

po[z]=temp1;

}

}

}

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

{

flag=1;

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

{

if(p[i]<=m[k])

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB allocated at memory partition:\t"<<po[k];

m[k]=0;

break;

}

else

{

flag++;

}

}

if(flag>n)

{

cout<<"\nprocess "<<i<<" whose memory size is "<<p[i]<<"KB can't be allocated";

}

}

break;

}

cout<<"\n Do you want to continue(y/n)";

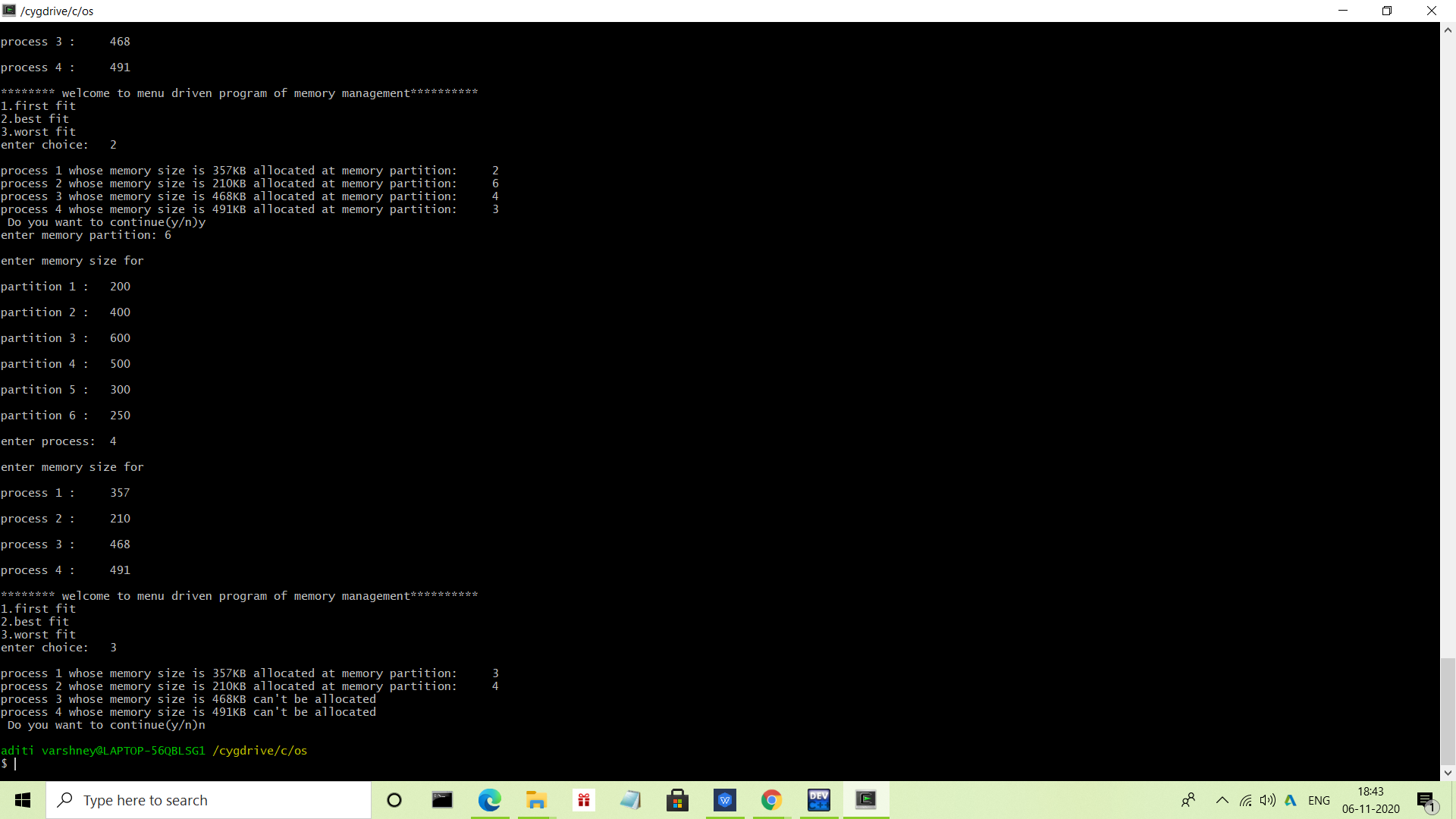
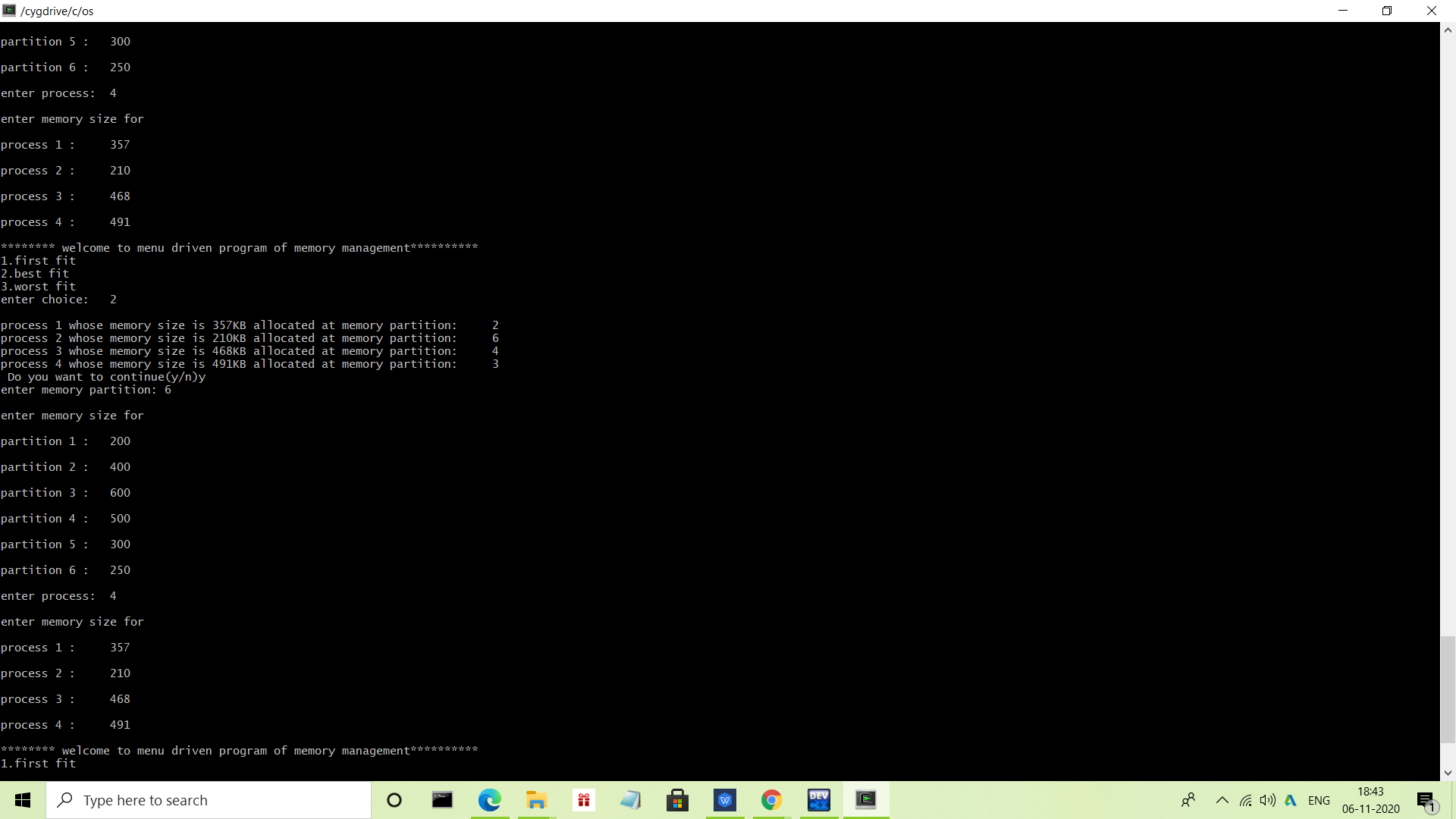
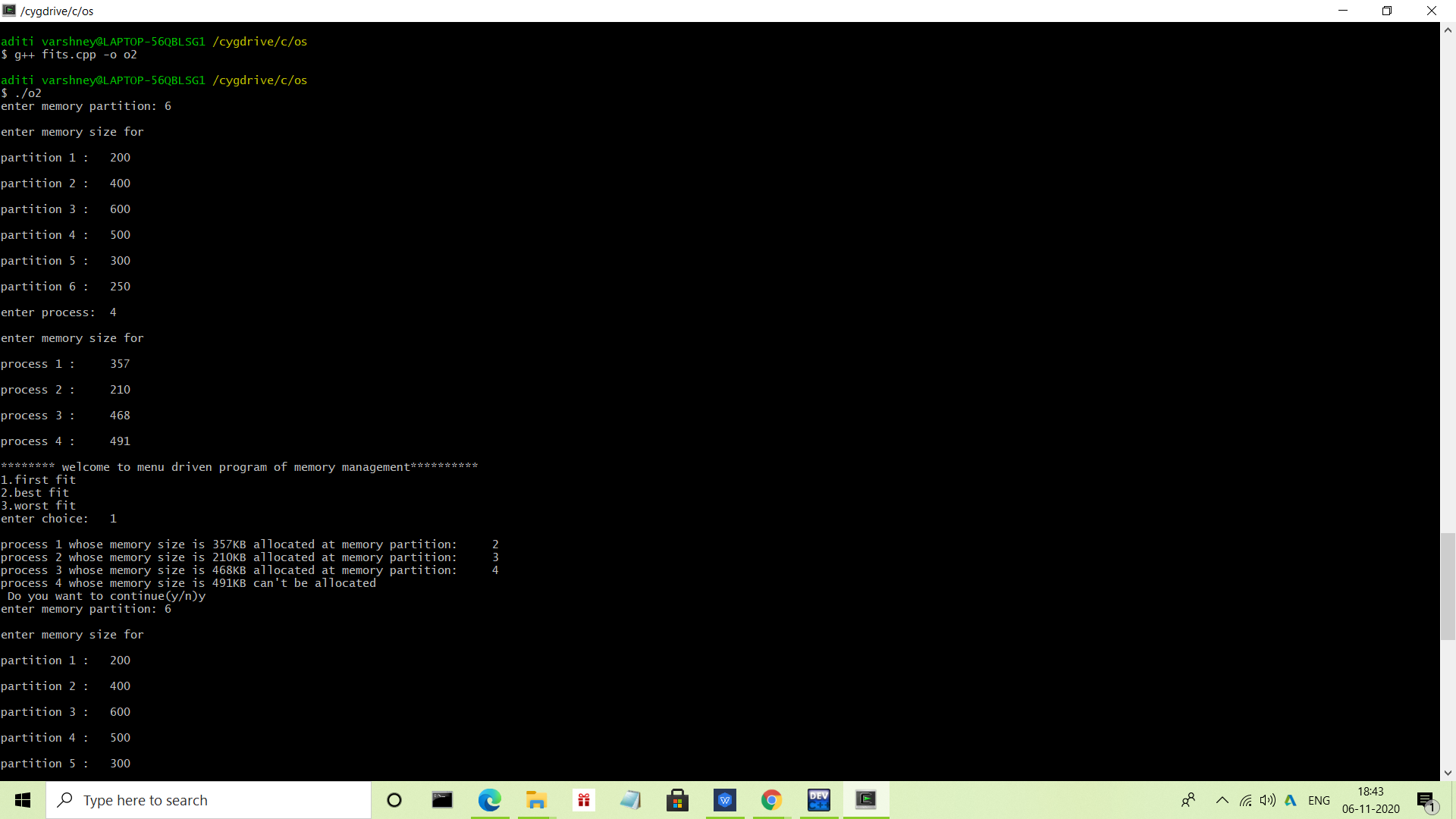
cin>>r;

}while(r=='y' || r=='Y');

getch();

return 0;

}



\*\*\*\*\*Thank You\*\*\*\*\*