ASSIGNMENT - III

NAME: SOUMYADIP GHOSH

STREAM: CSE-A

ROLL NUMBER: 1951007

SUBJECT: OS LAB

Question 1: Write a program that accepts an integer num as command line argument. In your program create two threads. One thread should report the num! (Factorial of num). The other thread should report the summation of all factorials till num. [eg: if the argument is 5, one thread calculates 5!=120, another thread calculates 1!+2!+3!+4!+5!=153]. Each process should report its own id and caller’s id. When a thread ends, it should display meaningful message.

Code:

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<pthread.h>

void fac(int num);

void sum(int num);

void\* threadfunc(void \*args){

printf("\nThread function responding");

}

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

{

int num= atoi(argv[1]);

int ret1,ret2;

pthread\_t t1,t2;

ret1= pthread\_create(&t1,NULL,&threadfunc,NULL);

ret2= pthread\_create(&t2,NULL,&threadfunc,NULL);

if(ret1==0){

printf("\nFirst Thread creation succesful");

printf("\nThread ID of caller of first thread: %ld",t1);

fac(num);

printf("\nFirst Thread ends here\n\n");

}

else{

printf("\nThread creation unsuccessful");

}

if(ret2==0){

printf("\nSecond Thread creation succesful");

printf("\nThread ID of caller of second thread: %ld",t2);

sum(num);

printf("\nSecond Thread ends here\n\n");

}

else{

printf("\nThread creation unsuccessful");

}

return 0;

}

void fac(int num){

int i, fac=1;

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

{

fac=fac\*i;

}

printf("\nFOR THE BLOCK FACTORIAL PROCESS:");

printf("\nThread ID of first thread: %ld",pthread\_self());

printf("\nFactorial of %d is:%d",num,fac);

}

void sum(int num){

int sum=0,i;

int arr[num+1];

arr[0]=1;

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

arr[i]=arr[i-1]\*i;

sum+=arr[i];

}

printf("\nFOR THE BLOCK SUM PROCESS:");

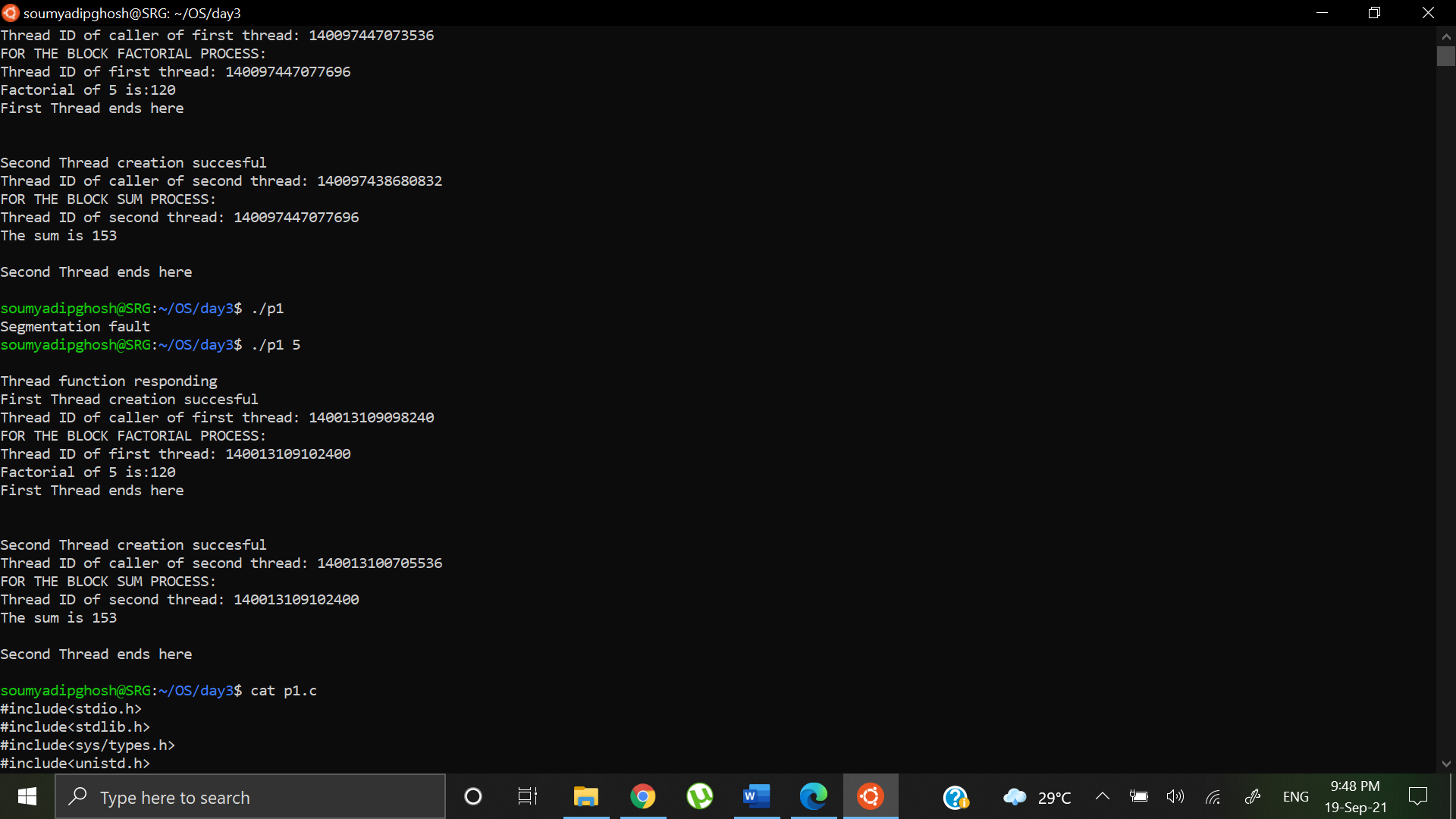
printf("\nThread ID of second thread: %ld",pthread\_self());

//printf("\nPID = %d, PARENT PID = %d\n",getpid(), getppid());

printf("\nThe sum is %d\n",sum);

}

Output:



Question 2: Write a program that accepts two integers (low, high) as command line argument. Create two threads in your program. The first thread should calculate the summation of all integers between (low, high) as sum\_res. The second thread should evaluate whether sum\_res is prime or not. Each process should report its own id and caller’s id. When a thread ends, it should display meaningful message.

Code:

#include<stdio.h>

#include<stdlib.h>

#include<sys/types.h>

#include<unistd.h>

#include<sys/wait.h>

#include<pthread.h>

int sum\_res=0,low,high;

void\* sumres(void\* args){

int i;

int \*ptr= &sum\_res;

pthread\_t \*id= (pthread\_t\*)args;

pthread\_t sid;

for(i=low;i<=high;i++){

\*ptr+=i;

}

sid= pthread\_self();

printf("\nThe sum is: %d",\*ptr);

printf("\nThread ID is %lu, Caller ID is %lu",sid,\*id);

pthread\_exit(NULL);

}

void \*primecheck(void\* args){

int i, flag=1;

int \*ptr= &sum\_res;

pthread\_t sid;

pthread\_t \*cid;

for (i=2; i<=\*ptr/2; i++){

if(\*ptr%i==0){

flag = 0;

break;

}

}

cid= (pthread\_t\*)args;

sid= pthread\_self();

sleep(0.5);

if (flag==1)

printf("\nsum\_res:%d is Prime",\*ptr);

else

printf("\nsum\_res: %d is Composite",\*ptr);

printf("\nThread ID is %lu, Caller ID is %lu\n",sid,\*cid);

}

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

int i,flag=1,sum\_res=0;

pthread\_t t1,t2;

low= atoi(argv[1]);

high= atoi(argv[2]);

t1= pthread\_create(&t1,NULL,sumres,&t1);

if(t1==0){

printf("\nThread 1 created succesfully");

}

else{

printf("\nThread creation unsuccesful");

}

pthread\_join(t1,NULL);

t2= pthread\_self();

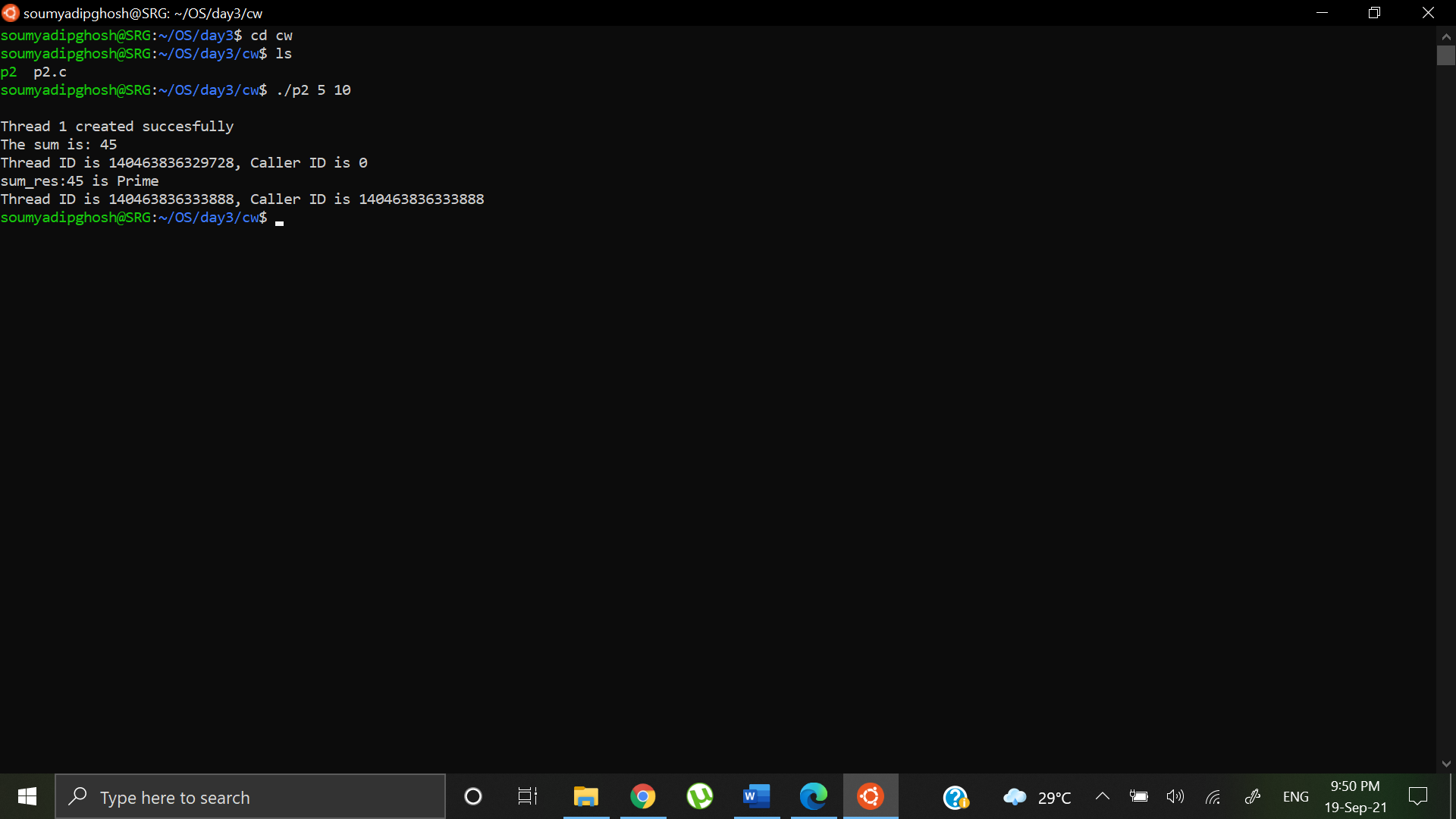
primecheck((void\*)t2);

pthread\_exit(NULL);

return 0;

}

Output:



Homework:

Question 1a: Write a multi-threaded program such that it creates 10 more threads. Each thread should print Hello from nth thread along with the argument received from the main thread. Each thread should return its own id and a unique value to main using pthread\_exit(). Main thread should be able to print this returned message identifying which thread ended within its own thread.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#include <string.h>

pthread\_t tid[10];

char \*ret[10];

// The function to be executed by all threads

void \*test\_thread(void \*arg)

{

unsigned long i = 0;

pthread\_t id = pthread\_self();

if(pthread\_equal(id,tid[0]))

{

printf("\n1 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[1]))

{

printf("\n2 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[2]))

{

printf("\n3 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[3]))

{

printf("\n4 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[4]))

{

printf("\n5 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[5]))

{

printf("\n6 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[6]))

{

printf("\n7 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[7]))

{

printf("\n8 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else if (pthread\_equal(id,tid[8]))

{

printf("\n9 created with ID: %lu", pthread\_self());

pthread\_exit(&id);

}

else

{

printf("\n10 created with ID: %lu\n\n", pthread\_self());

pthread\_exit(&id);

}

}

int main()

{

int i;

int \*ptr[10];

pthread\_t sid;

sid = pthread\_self();

printf("\nMAIN thread created with ID: %lu\n",sid);

for (i = 0; i < 10; i++) {

if (pthread\_create(&(tid[i]), NULL, test\_thread, (void \*)&tid)){

perror ("\npthread\_create() error");

exit(1);

}

else{

sleep(0.5);

}

}

for(int i=0;i<10;i++){

pthread\_join(tid[i],(void\*\*)&(ptr[0]));

}

for(int i=0;i<10;i++){

printf ("\n%d has ended with ID %lu ", i+1,tid[i]);

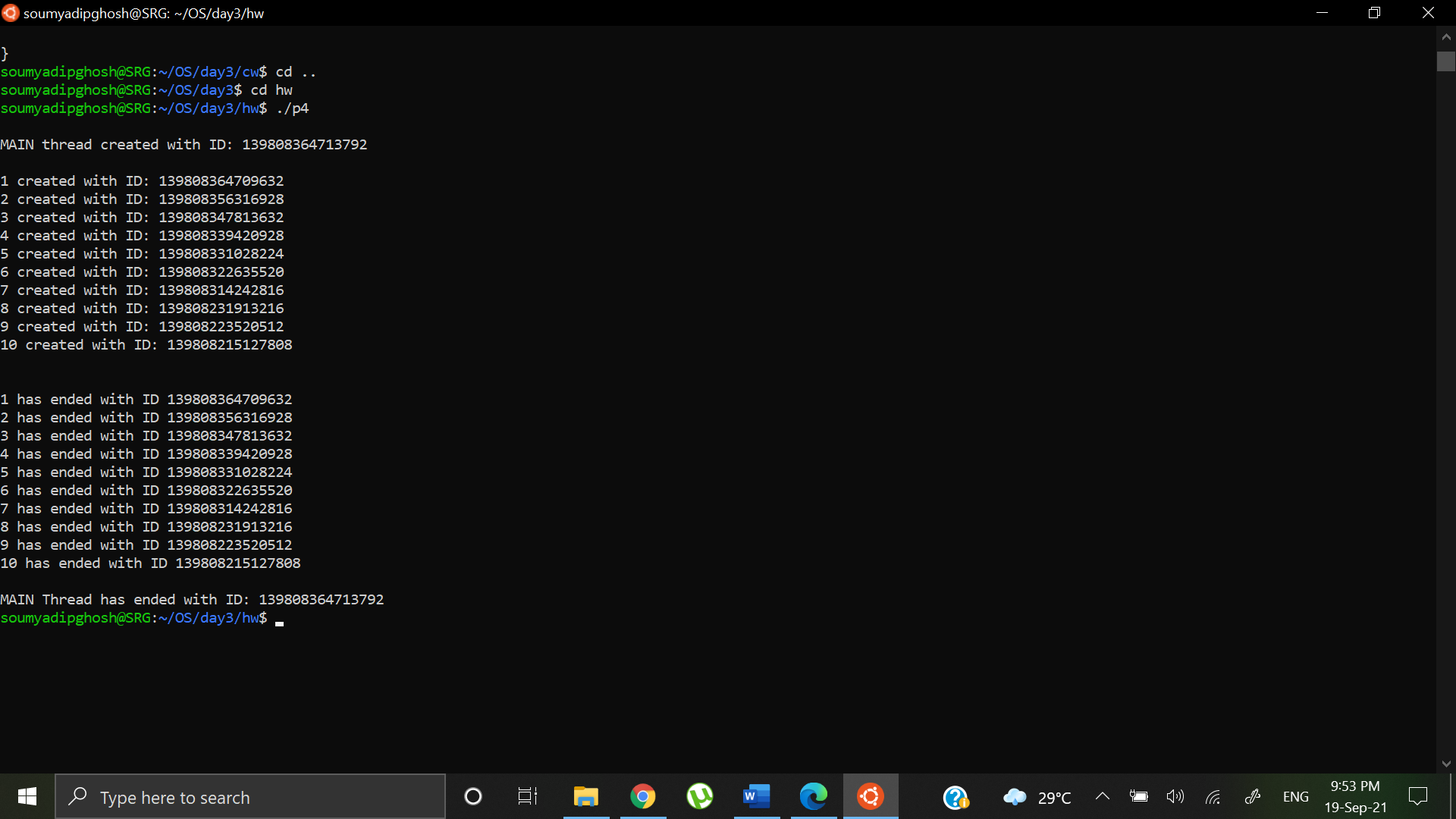
}

printf ("\n\nMAIN Thread has ended with ID: %lu\n" , pthread\_self());

return 0;

}

Output:



Question 1b: Synchronize it such that one is created and ends before the next one is created (till all 10 are created and terminated)

Code:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

// The function to be executed by all threads

void \*test\_thread(void \*arg)

{

// Store the value argument passed to this thread

pthread\_t sid;

sid= pthread\_self();

int id = \*(int\*)arg;

printf("\n%d created with ID: %lu",id,sid);

pthread\_exit((void \*)sid);

}

int main()

{

int i;

pthread\_t tid;

void \*rid= NULL;

for (i = 0; i < 10; i++) {

if (pthread\_create(&tid, NULL, test\_thread, (void \*)&i)){

perror ("\npthread\_create() error");

exit(1);

}

pthread\_join(tid,&rid);

printf("\n%d has ended with ID: %lu",i,(long)rid);

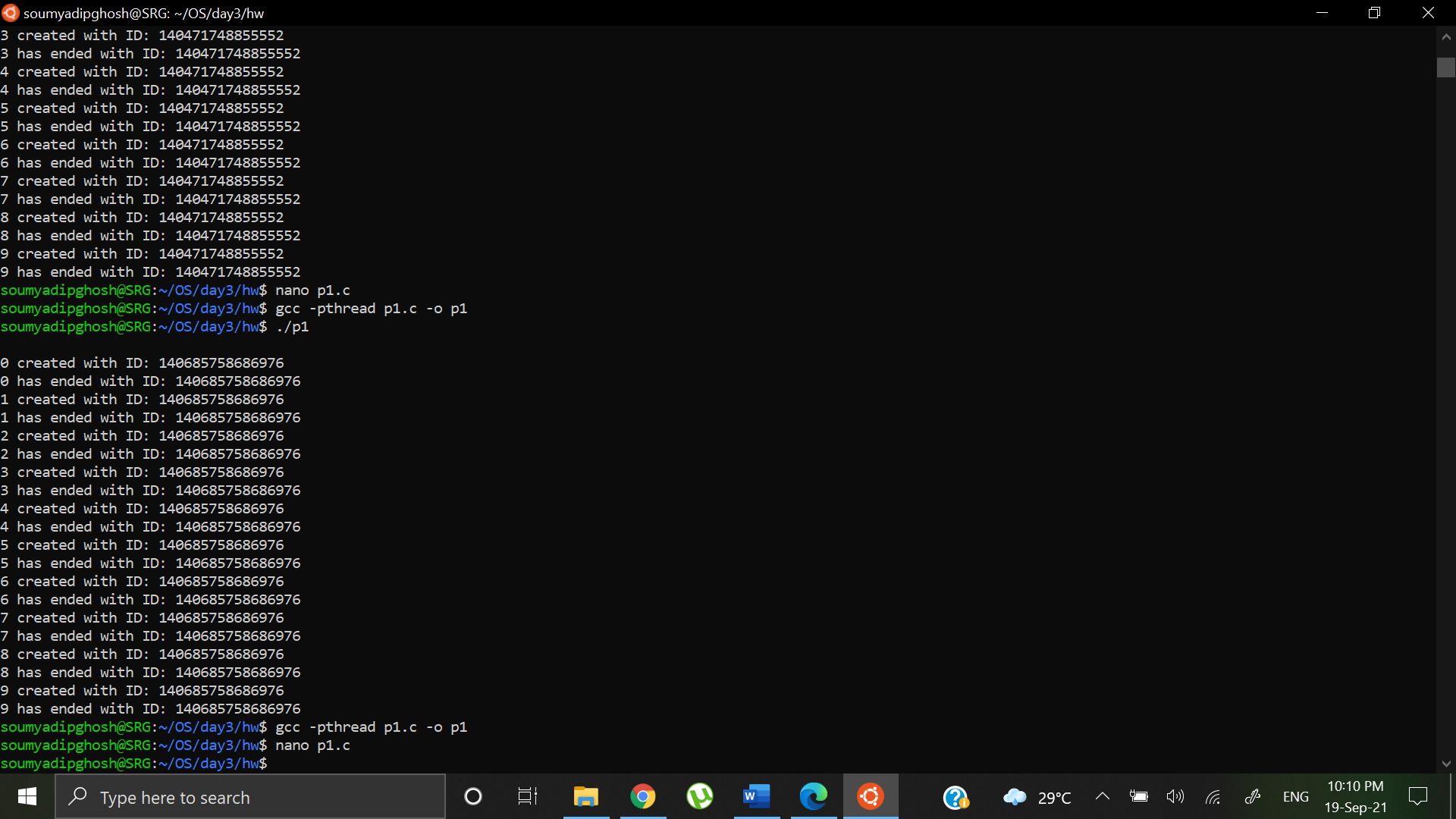
}

printf("\n");

return 0;

}

Output:



Question 1c: Synchronize it such that a pair is created at a time and terminated in the reverse order of how they were created

Code:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

// The function to be executed by all threads

void \*test\_thread(void \*arg)

{

pthread\_t sid= pthread\_self();

int id = \*(int\*)arg;

printf("\n%d created with ID: %lu",id,sid);

pthread\_exit(NULL);

}

int main()

{

int i,c1=1,c2=2;

pthread\_t tid1,tid2;

void \*rid;

for (i = 1; i <= 5; i++) {

if (pthread\_create(&tid1, NULL, test\_thread, (void \*)&c1)){

perror ("\npthread\_create() error");

exit(1);

}

if (pthread\_create(&tid2, NULL, test\_thread, (void \*)&c2)){

perror ("\npthread\_create() error");

exit(1);

}

pthread\_join(tid2,NULL);

printf("\n%d has ended with ID: %lu",c2,tid2);

pthread\_join(tid1,NULL);

printf("\n%d has ended with ID: %lu",c1,tid1);

//printf("\nEnd of %d iteration",i);

c1+=2;

c2+=2;

}

printf("\n");

return 0;

}

Output:

