PROBLEM:

Question no:5

A university CSE dept have Teaching assistant (TA) for clarifying doubts of students. In TA room there is only availability of one chair for student to sit and wait.Out side of TA room in hallway there are three chairs for students to sit and wait. TA will take rest whenever there is no student with doubt. if all seats are filled with students then new students will have to come again.

CONCEPTS USED:

Mutex lock for synchronizing the students entry and exit into TA room.

Threads used for processing simultaneous data or entry exit of student according to availability of TA time.

Semaphores is used for multitasking ,to abstract data type to control a common resource in a concurrent system.

CODE:

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

#include<pthread.h>

#include<string.h>

#include<semaphore.h>

void \*student\_activity(void\* student\_sno);

void\* TA\_activity();

semaphore\_m semaphore\_student;

semaphore\_m semaphore\_TA;

pthread\_mutex\_m mutex\_thread;

int waiting\_hall\_chairs[3];

int num\_of\_hall\_chairs=0;

int total\_waiting\_chairs=0;

int total\_students\_waiting=0;

int next\_seating=0;

int next\_teaching=0;

int TA\_sleep\_check=0;

int main(int m,char\*\*n)

{

int student\_count;

int total\_existing\_std;

int i;

if(isNumber(n[1]==1)

{

student\_count=atoi(n[1]);

}

else

{

printf("Invalid inp ,check again, retry");

return 0;

}

}

else

{

student\_count=total\_existing\_std;

}

int student\_sno[student\_count];

pthread\_m students[student\_count];

pthread\_m TA;

semaphoe\_init(&semaphore\_student,0,0);

semaphore\_init(&semaphore\_TA,0,1);

pthread\_mutex\_init(&mutex\_thread,NULL);

pthread\_creation(&TA,NULL,TA\_activity,NULL);

for(int i=0; i<student\_count;i++)

{

student\_sno[i]=i+1;

pthread\_creation(&students[i],NULL,student\_activity,(void\*)&student\_sno[i]);

}

pthread\_Add(TA,NULL);

{

for(int i=0; i<student\_sno;i++)

{

pthread\_Add(students[i],NULL);

}

return 0;

}

void\* TA\_Activity()

{

printf("checking students ");

while(1)

{

if(no\_of\_std\_wait>0)

{

TA\_sleep\_check=0;

semaphore\_wait(&semaphore\_students);

pthreads\_mutex\_lock(&mutex\_thread);

int official\_helping\_time=rand()%5;

printf("Helping hours are %d and student working hrs is % d", official\_helping\_time,(total\_students\_waiting -1);

printf("student %d reciving help ",waiting\_hall\_chair[next\_teaching\_pos]);

waiting\_hall\_chair[next\_teaching\_pos]=0;

total\_students\_waiting--;

next\_teaching\_pos=(next\_teaching\_pos+1)%Num\_waiting\_chairs;

sleep(official\_help\_time);

pthread\_mutex\_unlock(&mutex\_thread);

semaphore\_pos(&semaphore\_TA);

}

else

{

if(TA\_sleep\_check=0)

{

printf("no student waiting");

TA\_sleep\_check=1;

}

}

}

}

void\* student\_Activity(void\* student\_sno);

{

int sno\_student=\*(int\*)student\_sno;

while(1)

{

if(iswaiting(sno\_student)==1)

{

continue;

}

int time =rand()%5;

printf('student % d is programming for %d sec", sno\_student.time);

sleep(time)

pthread\_mutex\_lock(&mutex\_thread);

if(total\_student\_waiting<number\_of\_hall\_chairs)

{

waiting\_hall\_chair[next\_seating\_pos]=sno\_student;

total\_student\_wait ++

printf("students %d takes a seat student waiting %d", sno\_student,total\_students\_waiting);

next\_seat\_pos=(next\_seat\_pos+1)%total\_waiting\_chairs;

pthread\_mutex\_unlock(& mutex\_thread);

semaphore\_pos(&semaphore\_std);

semaphore\_wait(&semaphore\_TA);

}

else

{

pthread\_mutex\_unlock(&mutex\_thread);

printf("no chair is available, sorry come again");

}

}

}

int isNumber(char number[])

{

int i;

for(i=0;number[i]=0;i++)

{

if(is digit(number[i]))

{

return 0;

}

}

int iswaitng(int student\_sno)

{

for(int i=0; i<3; i++)

{

if(waiting\_hall\_chair[i]==student\_sno;

{

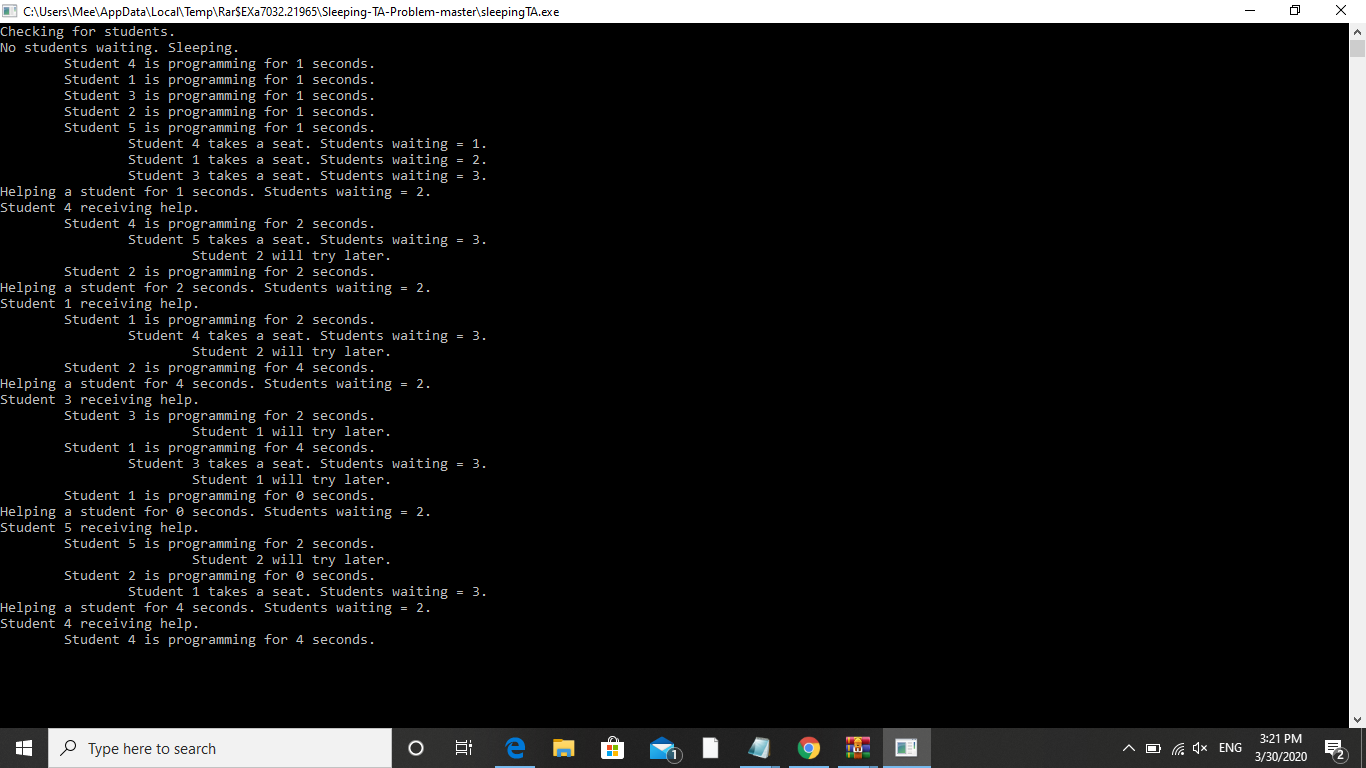
return 1;

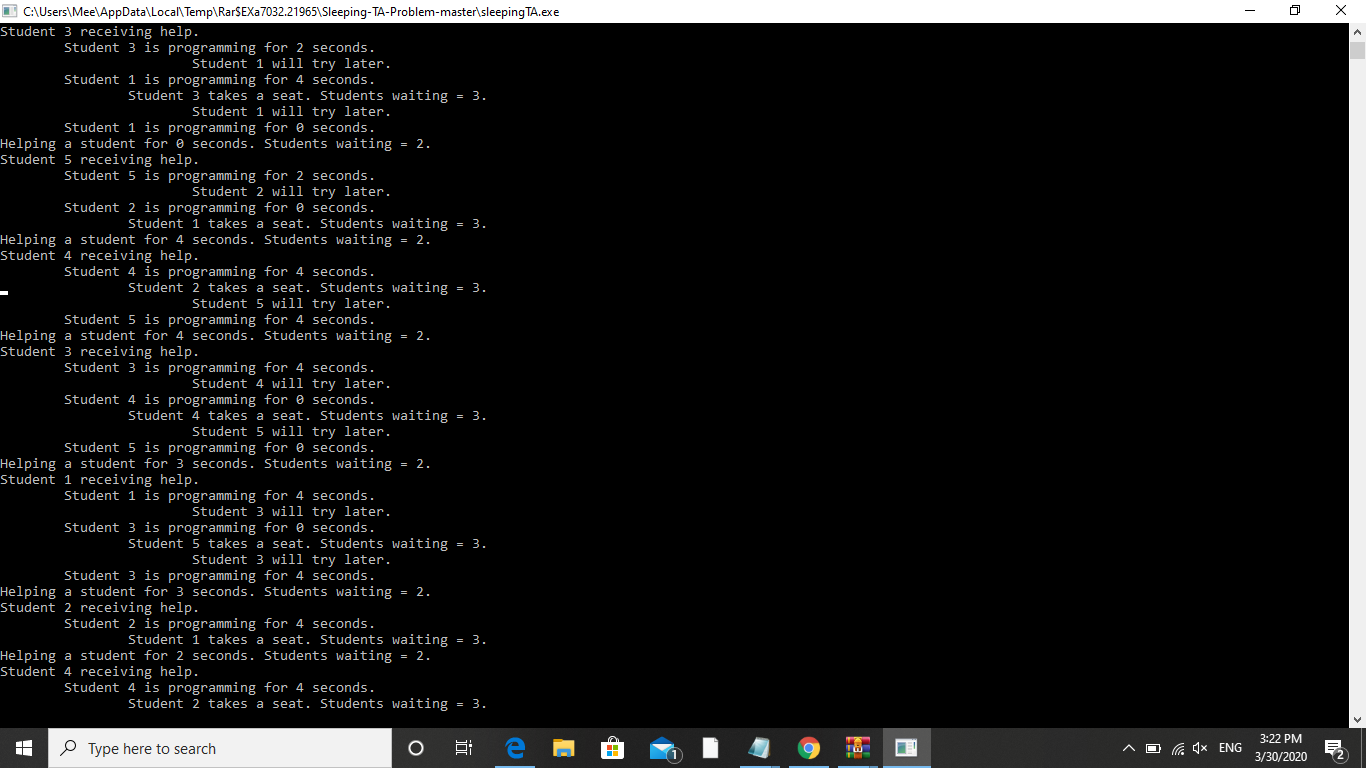
}

return 0;

}

OUTPUT:





ALGORITHM:

1.Declare constraints such as total\_number of students, students\_waiting, No of chairs present, Hallway\_chairs, Waiting list of students etc..

2.Create a thread for joining the student activity and teaching activity by importing pthread header file.With pthread\_.

3.Create a mutex lock with pthread to check whether Assistant sleeping or not.

4.If there is a mutex lock create a mutex unlock variable with pthread so that further monitoring will be possible.

5.Create a student\_activity function to monitor the student activities.

6.Create a function that describes Teaching assistant activities with name Assistant\_activity.

7.Create a semaphore variables so that we can access the activities of both student and teaching assistant activities.

8.Use necessary for …loops to check the activities.

9.Use while loop to check the Boolean function ture or not if true return the activity.

10.Finally print the necessary information that we need like activities of student, checking assistant sleeping or not, teaching assistant activity etc..

PURPOSE OF USE:

This project helps us to understand the application of semaphore,

Mutex lock & unlock, also gives us brief idea of using thread and creation of thread functions etc.

TEST CASES:

1.Checking assistant sleeping or not….This uses flag it returns false if he awake or it returns true if he sleeps.

If he sleeps the function terminates and wait until he awakes.

Or if he awakes function checks the assistant activity function and returns the value.

2.Student activity ..

If there is no availability of chairs then student waits .

If there is availability he occupies the chair and wait until he gets chance to meet assistant.

3.Teaching assistant activity..

If he sleeps his all other activity gets terminates..

If he awake then he teaches student with queue allotted to him..