Question-2 Report

Idea

In my idea all students are student threads, vaccination zones are vaccination zone threads and companies are company threads, students wait on the vaccination zones to get slots and vaccination zone waits on the companies for vaccines to produce. I have arrays and structures for each thing i.e for students i have st struct and s as array , for vaccination zone vz as struct and v as array and companies comp as struct and c as array.*I commented the code for more explaination.*

Implementation

required structs and arrays; comments are self explanatory for struct variables.

```
int m, n ,o;
typedef struct comp
{
    int id;
    long double prob;            //succes probability of vaccine
    int noofbatches;              //stores nof batches
    int noofvaccines;             //stores nof vaccines per batch
    int givestonoofvzones;         //stores for how many vaccination zones
it can give
    pthread_mutex_t lock;
}comp;
typedef struct vz
{
    int id;
    int vc;
    long double prob;            //succes probability of vaccine
    int noofvaccines;             //noof vaccines in vaccination zone
    int slots;                    //noof slots declared
    int noofslotsgiven;                //noof slots booked
    int comanyid;                   //storing company id which has given a
batch
    pthread_mutex_t lock;
}vz;
typedef struct st
{
    int id;
    int round;
    int result;
}st;
comp *c;
vz *v;
st *s;
long double p[10005];
int nos;
int ws;
```

```
pthread_mutex_t mutex;
```

other neccesary functions :

```
int myrandom(int l, int r)
{
    return (rand() % (r - l + 1)) + l;
}
int antib(long double prob)
{
    long double rr = (long double)rand() / RAND_MAX;
    if (rr > prob)
    {
        return 0;
    }
    else
    {
        return 1;
    }
}
int minimum(int a,int b,int c)
{
    if(a<=b && a<=c)
    {
        return a;
    }
    else if (b<=c && b<=a)
    {
        return b;
    }
    else
    {
        return c;
    }

}
```

In int main I am initialising the struct variables and creating threads for each student , each vaccination zone and each company,

```
c=malloc(n*sizeof(comp));
v=malloc(m*sizeof(vz));
s=malloc(o*sizeof(st));
nos=o;
ws=0;
printf("Give me the probabilities\n");
for(int i=0;i<n;i++)          //initialising companies
```

```
{
    c[i].id=i;
    c[i].noofbatches=0;
    c[i].noofvaccines=0;
    c[i].givestonoofvzones=0;
    scanf("%Lf",&c[i].prob);
    pthread_mutex_init(&c[i].lock,NULL);
}
pthread_mutex_init(&mutex,NULL);
for(int i=0;i<m;i++)           //initialising vaccine zones
{
    v[i].id=i;
    v[i].vc=0;
    v[i].noofslotsgiven=0;
    v[i].noofvaccines=0;
    v[i].prob=0;
    v[i].slots=0;
    v[i].comanyid=-1;
    pthread_mutex_init(&v[i].lock,NULL);
}
for(int i=0;i<o;i++)               //initialising students
{
    s[i].id=i;
    s[i].round=1;
    s[i].result=0;
}
printf("BeginSimulation\n");
pthread_t company_thread[n],vaccination_thread[m],student_thread[o];
for(int i=0;i<n;i++)
{
    pthread_create(&company_thread[i],NULL,fncompany,&c[i]);     //creating
thread for companies
}
for(int i=0;i<m;i++)
{
    pthread_create(&vaccination_thread[i],NULL,fnvzone,&v[i]);   //creating
thread for vaccinezones
}
for(int i=0;i<o;i++)
{
    pthread_create(&student_thread[i],NULL,fnstudent,&s[i]);
//creating thread for students
}
```

fncompany():

This function will be called by the company thread.the company have to prepare vaccines so the batches will be created randomly from 1 to 5 and noof vaccinesperbatch is created randomly betwee 10 to 20 after that the sleep is to time taken to prepare and after that we have to lock and assign the values of batches , vaccines and noofvzones it gives to in c array and unlock the lock, we will wait for the all batches taken by vzones to vaccinate and complete the vaccines then only the company will produce another time thats why i kept it in the while loop. In between if no of students become zero then company have to exit.

```
void *fncompany(void *args)
{
    int id=((comp*)args)->id;
    int noofbatches;
    int noofvaccinesperbatch;
    while(nos>0)                 //should goto infinite loop upto nos>0 as
a company is keep giving vaccines
    {
        noofbatches=myrandom(1,5);      //taking noof batches between 1 to
5 randomly
        noofvaccinesperbatch=myrandom(10,20);       //taking the vaccines
perbatch randomly form 10 to 20
        printf("Pharmaceutical Company with id %d is preparing %d batches
of vaccines which have success probability
%Lf\n",c[id].id,noofbatches,c[id].prob);
        sleep(myrandom(2,5));       //takes time to prepare
        printf("Pharmaceutical Company with id %d has prepared %d batches
of vaccines which have success probability %Lf\n", c[id].id, noofbatches,
c[id].prob);
        pthread_mutex_lock(&c[id].lock);
        c[id].noofbatches=noofbatches;               //setting the prepares
noof batches and noof vaccines per batch
        c[id].noofvaccines=noofvaccinesperbatch;
        c[id].givestonoofvzones=noofbatches;       //As the noof batches is
numner of vaccination zones taht company may give.
        pthread_mutex_unlock(&c[id].lock);

        while(c[id].givestonoofvzones!=0)           //waiting to be
complete all the batches given to the  vaccinationzones  by vaccination
        {
            if(nos==0)                              // in between nos becomes
zero then we have to return
            {
                return NULL;
            }
        }
        printf("All the vaccines prepared by Pharmaceutical Company with id
%d are emptied. Resuming production now.\n",c[id].id);
    }
    return NULL;
}
```

fnvzone():

This function is called by the vaccinationzone thread it runs till all the students are proccessed. The vaccination zone first have to take vaccines form company. the comments in the code explains the filling of vaccines by a company.

```
int id=((vz*)args)->id;
    int noofslots;
```

```
    while(nos>0)          //we have to execute upto nos>0
    {
         int i=0;
         while(nos>0)        //the vaccinations zones have to have vaccines
when nos>0
         {
             pthread_mutex_lock(&c[i].lock);  //lock to ensure that no two
vaccinezone effect the comapany
             if(c[i].noofbatches>0)             //if they have batches then
give one batch to vaccination zone
             {
                 c[i].noofbatches--;                //decrementing as one
batch is given to vaccinataion zone
                 pthread_mutex_lock(&v[id].lock);
                 v[id].noofvaccines=c[i].noofvaccines;       //taking the
vaccines in a batch of comapany to a vaccination zone
                 v[id].prob=c[i].prob;                      //taking the
succes probability of vaccine
                 v[id].comanyid=c[i].id;                         //storing
the comapany which gave vaccines to vaccination zone

                 pthread_mutex_unlock(&v[id].lock);
                 pthread_mutex_unlock(&c[i].lock);
                 printf("Pharmaceutical Company with id %d is delivering a
vaccine batch to Vaccination Zone with id %d which has success probability
%Lf\n", c[i].id, v[id].id, v[id].prob);
                 sleep(1);             //time takes to deliver
                 printf("Pharmaceutical Company with id %d has delivered
vaccines to Vaccination zone with id %d, resuming vaccinations now\n",
c[i].id, v[id].id);
                 break;
             }
             pthread_mutex_unlock(&c[i].lock);        //if this company
does not have batches i,e this if will not work so we have to un lock
             i=(i+1)%n;               //iterate through companies
         }
```

After filling the vaccines in vaccination zone now it has to declare slots for students to register the slot. we have to declare slots until noof vaccines in the zone to be completed. at first we will wait for the students to come. after that we will take the noof slots as randomly between 1 to minimum(8,v[id].noofvaccines,cpyws) here cpys is copy of value of ws while waiting. we will use lock for updating slots variable to noof slots and vaccination phase as 0 (not vaccinating). after we will busy wait for the slot to be zero or waiting students to be zero, then we will lock the part we are updating the noof slots given i.e all slots may not be taken at that time the waiting students may be zero so we will store no of slots students has booked.and turning into vaccination phase after that we will busy wait to complete vaccinating i.e noofslotsgiven becomes zero if in between noof students becomes zero then we have to exit. after all vaccinated in this vaccination zone i will check that noof vaccines are zero other wise we have to declare the slots again.= until noof vaccines becomes zero.If noof vaccines becomes zero then we will reset the vaccination zone and decrementing the giventonoofvzones variable in company to know that the

company used one batch, And again this vaccination zone will fill vaccines with comapany as noof students availabe.

```
while(v[id].noofvaccines>0)        //this slot declaration part if
noofvaccines in the vaccinaion zone are there.
{
    v[id].vc=0;                   //making sure that vaccination zone is
not in vaccination phase to declare slotes
    int cpyws;
    while((cpyws=ws)<=0)            //waiting for the students to come if
ws is 0 then we have to wait for more students to occur
    {
        if(nos==0)                  //while waiting if the noof studnets =0
then we have to return
        {
            return NULL;
        }
    }
    noofslots=myrandom(1,minimum(8,v[id].noofvaccines,cpyws));  //declaring
slots according to noofvaccines that vaccinationzone have and waiting
students
    pthread_mutex_lock(&v[id].lock);
    v[id].slots=noofslots;
    printf("Vaccination Zone with id %d is ready to vaccinate with %d
slots\n",v[id].id,noofslots);
    v[id].vc=0;                  //making sure that the vaccination zone in
not in vaccination phase as studnets has to book slots
    pthread_mutex_unlock(&v[id].lock);
    while(v[id].slots>0 && ws>0)     // waiting for slots to be booked upto
slots >0 and noof waiting students >0
    {

    }
    pthread_mutex_lock(&v[id].lock);
    v[id].noofslotsgiven = noofslots-v[id].slots;       //setting no of
slots given i.e noof slots booked
    v[id].vc = 1;                                  //entering into
vaccintion phase
    printf("Vaccination Zone with id %d entering Vaccination
Phase\n",v[id].id);
    pthread_mutex_unlock(&v[id].lock);
    while(v[id].noofslotsgiven>0)                //waiting to vaccinate the
slot booked students.
    {
       if(nos==0)                           //while in between nos becomes
0 then return
       {
           return NULL;
       }
    }
    pthread_mutex_lock(&v[id].lock);
    if(v[id].noofvaccines>0)                    //checking that the
```

```
vaccines are over or not in vaccination zone
    {                                           //if not over then we have
to make slots for the rest of vaccines left
        v[id].vc=0;                             //setting it to not
vaccinating phase
        v[id].noofslotsgiven=0;                 //setting noof slots and
booked slots to 0
        v[id].slots=0;
        pthread_mutex_unlock(&v[id].lock);
        continue;                               //and continue to again
slot declaration
    }
    else
    {                                          //if they are over then we
will make sure that all will zero
        v[id].noofslotsgiven=0;
        v[id].noofvaccines=0;
        v[id].prob=0;
        v[id].slots=0;
        v[id].vc=0;
        printf("Vaccination Zone with id %d has run out of
vaccines\n",v[id].id);
        pthread_mutex_unlock(&v[id].lock);
        pthread_mutex_lock(&c[v[id].comanyid].lock);
        c[v[id].comanyid].givestonoofvzones--;     //decrasing the
noofgivenvzones in company
        pthread_mutex_unlock(&c[v[id].comanyid].lock);
        pthread_mutex_lock(&v[id].lock);
        v[id].comanyid=-1;                          //setting the
vaccinationzone's company to -1 i.e no company is assigned to this vzone
        pthread_mutex_unlock(&v[id].lock);
    }
}
```

fnstudent():

This funtion will be called by student thread. The student will arrive late so we sleep for sec between 1 to 5 randomly.Then we enter the while loop if the students result is 0(tested negative in antibody test) and the student round is <=3. As we enter the loop we will increase waiting students and we will iterate through the vaccination zones which has slots and it is not in vaccination phase. after satisfying these the student will enter and decrease the slots and waiting students. and they will busy wait until the vaccination phase begins. After vaccination phase begins we have to decrement the noof vaccines and noofslotsgiven because the students are vaccinating we will do sleep 1 after this becasue he is taking time for vaccination after vacinating the students will take antibody test i.e a fn which takes the probability of vaccine to success and make random value and checking with probabilty if random value > probabilty then it is succeded(i.e psoitive) otherwise failed (i.e negative) after that we are sleeping i.e time taken to test . after that we will check the result . if result is 0 then he is negative and doing round++ i.e going vaccination for another round other wise student is vaccinated and getts out of the loop and if the student got out of the loop and his result is 0 then he has to go to home otherwise going to school any ways the students are gone form vaccinating so noof students is decremented and exiting.

```
void *fnstudent(void *args)
{
    sleep(myrandom(1,5));          //sleeping between 1,5 is the students
not coming at time
    int id=((st*)args)->id;
    while(s[id].result==0 && s[id].round<=3)   //if students round should
be less than 3 and his result should be negative i.e 0 for to enter this
vaccinations
    {
        printf("Student with id %d has arrived for his %d round of
Vaccination\n",s[id].id,s[id].round);
        printf("Student with id %d is waiting to be allocated a slot on a
Vaccination Zone\n",s[id].id);
        int i=0;
        pthread_mutex_lock(&mutex);               // locking because ws is
the critical section for student
        ws++;                                  //increasing the stundents
who are waiting
        pthread_mutex_unlock(&mutex);
        while(nos)                            //while no of students >0
        {
            pthread_mutex_lock(&v[i].lock);          //lock to avoid
multiple students to be accessed at same time
            if(v[i].slots>0 && v[i].vc==0)         //if slots are there and
the vaccination zone is not in vaccination phase then we have to declare
slots
            {
                pthread_mutex_lock(&mutex);    //lock as this the critical
section for students
                ws--;                        //as slot is given to student
then the student is not waiting for the slot
                pthread_mutex_unlock(&mutex);
                v[i].slots--;                  //as slot is given to student
then slots should be decreased.
                printf("Student with id %d assigned a slot on the
Vaccination Zone with id %d and waiting to be
vaccinated\n",s[id].id,v[i].id);
                pthread_mutex_unlock(&v[i].lock);  //unlocking the
vaccination lock
                while(v[i].vc==0)                  //waiting for the zone
to enter vaccination phase
                {

                }
                pthread_mutex_lock(&v[i].lock);  //lock as multiple
students cannot acces the thing in same time
                v[i].noofvaccines--;             //as the students is
vaccinates the the noofvacines should be decrement
                v[i].noofslotsgiven--;           //so the actual slots
given to students to be drecremented (v[i].slots is not actual slots the
actual slots booked is less than or equal to this so we have to decrement
this)
```

```
                pthread_mutex_unlock(&v[i].lock);
                sleep(1);                   //sleep the student to be
vaccinated as vaccination takes some time.
                printf("Student with id %d on Vaccination Zone with id %d
has been vaccinated which has success probability %Lf\n", s[id].id,
v[i].id, v[i].prob);
                pthread_mutex_lock(&mutex);
                s[id].result=antib(v[i].prob);  //checking that the
students has tested positive or negative for antibodies
                sleep(1);
                if(s[id].result==0)
                {
                    printf("Student with id %d has tested negative for
antibodies.\n",s[id].id);
                    s[id].round++;  //as it is negative so we have to
increment his round to vaccinate again
                    pthread_mutex_unlock(&mutex);
                    break;
                }
                else
                {
                    printf("Student with id %d has tested positive for
antibodies.\n",s[id].id);
                    pthread_mutex_unlock(&mutex);
                    break;
                }
            }
            pthread_mutex_unlock(&v[i].lock);   //unlocking the vaccination
lock if the if statement is not worked
            i=(i+1)%m;                //iterate through vaccination zones
        }
    }
    if(s[id].result==0)        //if the result is zero after rthe while
loop then he should go to homw
    {
        printf("student with id %d should go to home for another online
semester\n", s[id].id);
    }
    pthread_mutex_lock(&mutex); // as nos is critical section for students
    nos--;                     // as the students completed his vaccinatio
no of students should be reduced
    pthread_mutex_unlock(&mutex);
    return NULL;

}
```

we have to wait for all the threads to join and destroy the locks:

```
for (int i = 0; i < n; i++)
    {
        pthread_join(company_thread[i], NULL);
```

```
    }
    for (int i = 0; i < m; i++)
    {
        pthread_join(vaccination_thread[i], NULL);
    }
    for (int i = 0; i < o; i++)
    {
        pthread_join(student_thread[i], NULL);
    }

    for (int i = 0; i < n; i++)
    {
        pthread_mutex_destroy(&c[i].lock);
    }
    for (int i = 0; i < m; i++)
    {
        pthread_mutex_destroy(&v[i].lock);
    }
    pthread_mutex_destroy(&mutex);

    printf("Simulation Over\n");
```