Assignment-4

Question-2

Back to College

Risubrik is a fellow student of an IIIT-H, and he is completely red of the online semester and wants to head back to college as soon as possible. He wants to suggest that the college get all the students vaccinated using his huge internship money and a foolproof plan to do so. Help him devise a plan that he can impress the college with.

Global Structs

```
typedef  struct  vaccination_zone
{
    int  id;
    int  capacity;
    int  batchCompany;
    int  NumSlots;
    bool  ready;
    pthread_t  tid;
    pthread_mutex_t  m;
} vaccination_zone;

typedef  struct  company
{
    int  id;
    int  NumBatches;
    pthread_t  tid;
    pthread_mutex_t  m;
} company;

typedef  struct  student
{
    int  id;
    int  uid;
    int  cid;
    bool  vaccinated;
    pthread_t  tid;
    pthread_mutex_t  m;
} student;
```

Explanation

There are two global structs -- because threads share global address space, this is the easiest way to handle them. Each struct has its mutex to ensure that a single thread can access variables in the struct at a time.

Students

Student function

```
static  void  *Student(void  *s1)
{
    student  *s  = (student  *)s1;
    sleep(getBrowseTime());
    int  numTests  =  0;
    while (numTests  <  3)
    {
        printf("\033[1;33mStudent #%d has arrived for his #%d round of 
vaccination\n\n", s->id  +  1, numTests  +  1);
        fflush(stdout);
        pthread_mutex_lock(&lock);
        WaitingStudents++;
        pthread_mutex_unlock(&lock);
        getSlot(s);
        bool  test  =  testStudent(s->cid);
        const  char  *verb  =  test  ==  0  ?  "NEGATIVE"  :  "POSITIVE";
        printf("\033[1;33mStudent #%d has tested %s for antibodies\n\n", s-
>id  +  1, verb);
        fflush(stdout);
        if (test)
        {
            pthread_mutex_lock(&lock);
            VaccinatedStudents++;
            pthread_mutex_unlock(&lock);
            return  NULL;
        }
        numTests++;
    }
    pthread_mutex_lock(&lock);
    VaccinatedStudents++;
    pthread_mutex_unlock(&lock);
    return  NULL;
}
```

Explanation

This function handles all the details of a student. It uses a mutex to access the shared global variables to be no race condition and deadlock. It calls **getSlot** function to get a slot in one of the vaccination zones available at that time. After three consecutive failures, this function will exit, and the student is sent home for another online semester.

getSlot function

```
static  void  getSlot(student  *s)
{
    printf("\033[1;33mStudent #%d is waiting to be allocated a slot on a
vaccination zone\n\n", s->id  +  1);
    fflush(stdout);
    bool  success  =  false;
    while (!success)
    {
        for (int  i  =  0; i  <  NumVaccinationZones; i++)
        {
            pthread_mutex_lock(&vaccination_zones[i].m);
            if (vaccination_zones[i].NumSlots  >  0  &&
vaccination_zones[i].ready)
            {
                printf("\033[1;33mStudent #%d assigned a slot on the
vaccination zone #%d and waiting to be vaccinated\n\n", s->id  +  1,
vaccination_zones[i].id  +  1);
                fflush(stdout);
                s->uid  =  vaccination_zones[i].id;
                vaccination_zones[i].NumSlots--;
                success  =  true;
                pthread_mutex_unlock(&vaccination_zones[i].m);
                break;
            }
            pthread_mutex_unlock(&vaccination_zones[i].m);
        }
    }
    s->vaccinated  =  false;
    while (!s->vaccinated)
    {
        sleep(1);
    }
}
```

Explanation

In this function, a student is assigned to a vaccination zone; then, the student thread is waiting with short naps until it gets vaccinated.

Vaccination Zone

Vaccination zone function

```
static  void  *Vaccination(void  *v1)
{
    vaccination_zone  *v  = (vaccination_zone  *)v1;
    while (true)
    {
        if (v->capacity  ==  0)
        {
            printf("\033[1;32mVaccination zone #%d has run out of
vaccines\n\n", v->id  +  1);
            fflush(stdout);
            getBatch(v);
        }
        pthread_mutex_lock(&lock);
        int  slots  =  getSlotSize();
        v->NumSlots  =  minimum(slots, WaitingStudents, v->capacity);
        if (VaccinatedStudents  ==  NumStudents)
        {
            pthread_mutex_unlock(&lock);
            return  NULL;
        }
        WaitingStudents  -=  v->NumSlots;
        v->capacity  -=  v->NumSlots;
        pthread_mutex_unlock(&lock);
        if (v->NumSlots  >  0)
        {
            allocateSlots(v);
            giveVaccination(v);
        }
    }
}
```

Explanation

- This function handles all the details of a vaccination zone.
- If the vaccinations batch gets over, it calls the **getBatch** function to receive a batch from a pharmaceutical company available at that time.
- It will then find the number of slots it will allocate in this iteration and decrease the **WaitingStudents** global variable shared between different threads.
- If all the students are vaccinated, it will exit.

getBatch function

```
static  void  getBatch(vaccination_zone  *v)
{
    if (v->batchCompany  >=  0)
    {
        pthread_mutex_lock(&companies[v->batchCompany].m);
        companies[v->batchCompany].NumBatches--;
        pthread_mutex_unlock(&companies[v->batchCompany].m);
    }
    v->batchCompany  =  -1;
    while (v->batchCompany  ==  -1)
    {
        sleep(1);
    }
}
```

Explanation

It is a straightforward function waiting for a pharmaceutical company to send a batch of vaccinations to this calling vaccination zone. It also signals the previous batch company that its provided batch is over to resume production. The **allocateSlot** function is similar to this function.

giveVaccination function

```
static void giveVaccination(vaccination_zone *v)
{
    printf("\033[1;32mVaccination zone #%d entering vaccination phase\n\n",
v->id + 1);
    fflush(stdout);
    for (int i = 0; i < NumStudents; i++)
    {
        pthread_mutex_lock(&students[i].m);
        if (students[i].uid == v->id && !students[i].vaccinated)
        {
            students[i].cid = v->batchCompany;
            students[i].uid = -1;
            sleep(1); // to make the simulation realistic
            printf("\033[1;32mStudent #%d on vaccination zone #%d has been
vaccinated which has success probability %0.2lf\n\n", students[i].id + 1,
v->id + 1, VaccineSuccessProb[v->batchCompany]);
            fflush(stdout);
            students[i].vaccinated = true;
        }
        pthread_mutex_unlock(&students[i].m);
    }
}
```

Explanation

This function handles the details when a vaccination zone is in the vaccination phase. It will give vaccinations to the students who are assigned to this particular vaccination zone.

Pharmaceutical Company

Company function

```
static void *Company(void *c1)
{
    company *c = (company *)c1;
    bool success = true;
    while (success)
    {
        success  =  prepBatches(c);
        success  =  allocateBatch(c);
    }
    return  NULL;
}
```

Explanation

It is a straightforward function which calls two functions **prepBatches** and **allocateBatches**.

prepBatches function

```
static  bool  prepBatches(company  *c)
{
    c->NumBatches  =  getNumBatches();
    int  prep_time  =  getPrepTime();
    if (VaccinatedStudents  ==  NumStudents)
    {
        return  false;
    }
    printf("\033[1;34mPharmaceutical company #%d is preparing #%d batches
of vaccines which have success probability %0.2lf\n\n", c->id  +  1, c-
>NumBatches, VaccineSuccessProb[c->id]);
    fflush(stdout);
    sleep(prep_time);
    printf("\033[1;34mPharmaceutical company #%d has prepared #%d batches
of vaccines which have success probability %0.2lf\n\n", c->id  +  1, c-
>NumBatches, VaccineSuccessProb[c->id]);
    fflush(stdout);
    return  true;
}
```

Explanation

This function models the **real** pharmaceutical company. It is preparing some random number of batches taking some unexpected time until it swaps itself off the CPU.

allocateBatches function

```
static  bool  allocateBatch(company  *c)
{
    int  totalBathches  =  c->NumBatches;
    while (totalBathches  >  0)
    {
        for (int  i  =  0; i  <  NumVaccinationZones  &&  totalBathches  >
0; i++)
```

```
        {
            if (VaccinatedStudents  ==  NumStudents)
            {
                return  false;
            }
            vaccination_zone  *v  =  &vaccination_zones[i];
            pthread_mutex_lock(&v->m);
            if (v->batchCompany  ==  -1  &&  v->capacity  ==  0)
            {
                printf("\033[1;34mPharmaceutical company #%d is delivering
a vaccine batch to vaccination zone #%d which has success probability
%0.2lf\n\n", c->id  +  1, v->id  +  1, VaccineSuccessProb[c->id]);
                fflush(stdout);
                sleep(getBrowseTime());
                v->capacity  =  getVaccinationCapacity();
                printf("\033[1;34mPharmaceutical company #%d has delivered
vaccines to vaccination zone #%d, resuming vaccinations now\n\n", c->id  +
1, v->id  +  1);
                fflush(stdout);
                v->batchCompany  =  c->id;
                totalBathches--;
            }
            pthread_mutex_unlock(&v->m);
        }
    }
    while (c->NumBatches  >  0)
    {
        if (VaccinatedStudents  ==  NumStudents)
        {
            return  false;
        }
        sleep(1);
    }
    printf("\033[1;34mAll the vaccines prepared by pharmaceutical company
#%d are emptied, resuming production now\n\n", c->id  +  1);
    fflush(stdout);
    return  true;
}
```

Explanation

- This function is sending vaccination batch to different vaccination zones and signals them.
- It waits using short naps until all vaccination batches are used (signaled by vaccination zones by decrementing the variable **NumBatches**).

Analysis

I analyzed the code for a random small test case. Input and output are given below.

Input

```
Enter the number of pharmaceutical companies: 3
Enter the number of vaccination zones: 3
Enter the number of students to be vaccinated: 3
Enter the probabilities of success of each pharmaceutical company:
0.8 0.9 1
```

Output

```
Pharmaceutical company #1 is preparing #4 batches of vaccines which have
success probability 0.80

Pharmaceutical company #2 is preparing #3 batches of vaccines which have
success probability 0.90

Pharmaceutical company #3 is preparing #2 batches of vaccines which have
success probability 1.00

Vaccination zone #1 has run out of vaccines

Vaccination zone #2 has run out of vaccines

Vaccination zone #3 has run out of vaccines

Pharmaceutical company #3 has prepared #2 batches of vaccines which have
success probability 1.00

Pharmaceutical company #3 is delivering a vaccine batch to vaccination zone
#1 which has success probability 1.00

Student #1 has arrived for his #1 round of vaccination

Student #1 is waiting to be allocated a slot on a vaccination zone

Pharmaceutical company #1 has prepared #4 batches of vaccines which have
success probability 0.80

Pharmaceutical company #2 has prepared #3 batches of vaccines which have
success probability 0.90

Pharmaceutical company #3 has delivered vaccines to vaccination zone #1,
resuming vaccinations now

Pharmaceutical company #3 is delivering a vaccine batch to vaccination zone
#2 which has success probability 1.00

Vaccination zone #1 is ready to vaccinate with #1 slots

Pharmaceutical company #3 has delivered vaccines to vaccination zone #2,
resuming vaccinations now
```

```
Student #1 assigned a slot on the vaccination zone #1 and waiting to be
vaccinated

Pharmaceutical company #1 is delivering a vaccine batch to vaccination zone
#3 which has success probability 0.80

Vaccination zone #1 entering vaccination phase

Student #1 on vaccination zone #1 has been vaccinated which has success
probability 1.00

Student #2 has arrived for his #1 round of vaccination

Student #2 is waiting to be allocated a slot on a vaccination zone

Vaccination zone #2 is ready to vaccinate with #1 slots

Student #1 has tested POSITIVE for antibodies

Pharmaceutical company #1 has delivered vaccines to vaccination zone #3,
resuming vaccinations now

Student #2 assigned a slot on the vaccination zone #2 and waiting to be
vaccinated

Vaccination zone #2 entering vaccination phase

Student #2 on vaccination zone #2 has been vaccinated which has success
probability 1.00

Student #2 has tested POSITIVE for antibodies

Student #3 has arrived for his #1 round of vaccination

Student #3 is waiting to be allocated a slot on a vaccination zone

Vaccination zone #3 is ready to vaccinate with #1 slots

Student #3 assigned a slot on the vaccination zone #3 and waiting to be
vaccinated

Vaccination zone #3 entering vaccination phase

Student #3 on vaccination zone #3 has been vaccinated which has success
probability 0.80

Student #3 has tested NEGATIVE for antibodies

Student #3 has arrived for his #2 round of vaccination

Student #3 is waiting to be allocated a slot on a vaccination zone

Vaccination zone #3 is ready to vaccinate with #1 slots
```

```
Student #3 assigned a slot on the vaccination zone #3 and waiting to be
vaccinated

Vaccination zone #3 entering vaccination phase

Student #3 on vaccination zone #3 has been vaccinated which has success
probability 0.80

Student #3 has tested POSITIVE for antibodies

Simulation over, thank you!
```

Thank you!