Back to college

Breakdown of the implemented Code

Main highlights of Code

*Main Function *Company Function *Student Function *Zone Function

Structure and Global Variables used

The structure for company , zone , student is defined . Also the global variable like condition for the use in while loop and length of vaccine buff and student buff is defined.

```
ll  condition = 1, len_vac_buff = 0, len_stud_buff = 0; pthread_mutex_t vac_buff_mutex,
stud_buff_mutex;

 typedef struct company
 {
ll      num, batches, vac_per_bh; pthread_t comp_id;
     float prob;
     pthread_mutex_t comp_mutex;
 } company;

 company *vac_buff[VSIZE];

 typedef struct zone
 {
ll      num, slots, doses_left; pthread_t zone_id; pthread_mutex_t zone_mutex; company *comp;
 } zone;

 typedef struct student
 {
ll      num, tries, status, vacc; pthread_t stu_id; pthread_mutex_t stu_mutex;
 } student;

 student *stu_buff[BSIZE];
```

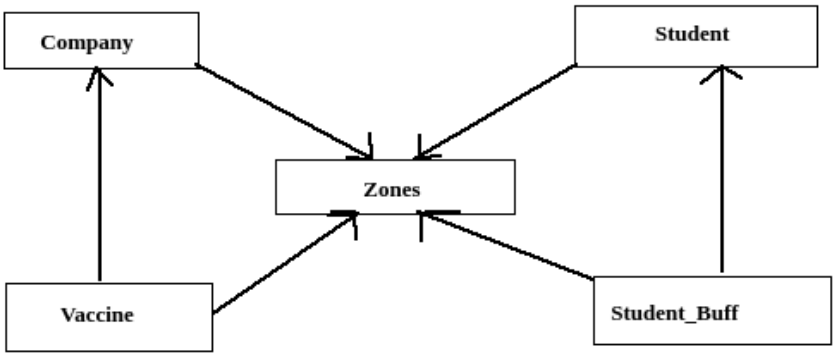
main() Function

Mutex of student buffer is initialized , created , joined and destroyed and the corresponding functions have been called.

```
int main()
{
    srand(time(0));
    .
    .
    pthread_mutex_init(&stud_buff_mutex, NULL);
    pthread_mutex_init(&vac_buff_mutex, NULL);

    printf("Enter Probabilities (Space Seperated)\n");
    for (ll i = 0; i <= (n - 1); ++i) {

        scanf("%f", &comp[i].prob);
        ll f = 0;
        if (comp[i].prob < 0 || comp[i].prob > 1)
        {
            f = 1;
            printf("Enter Correct Probability. Try again\ncondition Over\n");
        }
        if (f)
            return 0;
    }
    for (ll i = 0; i <= (n - 1); ++i)
    {
        int index = i + 1;
        pthread_mutex_init(&(comp[i].comp_mutex), NULL);
        comp[i].batches = 0;
        comp[i].num = index;
        pthread_create(&(comp[i].comp_id), NULL, comp_func, (void *)(&comp[i]));
    }

    for (ll i = 0; i <= (m - 1); ++i)
    {
        int index = i + 1;
        zn[i].num = index;
        pthread_mutex_init(&(zn[i].zone_mutex), NULL);
        zn[i].slots = 0;
        zn[i].doses_left = 0;
        pthread_create(&(zn[i].zone_id), NULL, zone_func, (void *)(&zn[i]));
    }

    for (ll i = 0; i <= (o - 1); ++i)
    {
        int index = i + 1;
        sleep(1);
        st[i].num = index;
        st[i].vacc = 0;
        pthread_mutex_init(&(st[i].stu_mutex), NULL);
        st[i].status = 0;
        st[i].tries = 0;
        pthread_create(&(st[i].stu_id), NULL, student_func, (void *)(&st[i]));
    }
```

```
    for (ll i = 0; i <= (o - 1); ++i)
    {
        pthread_join(st[i].stu_id, NULL);
    }
    for (ll i = 0; i <= (o - 1); ++i)
    {
        pthread_mutex_destroy(&(st[i].stu_mutex));
    }
    .
    .
    for (ll i = 0; i <= (n - 1); ++i)
    {
        pthread_join(comp[i].comp_id, NULL);
    }
    for (ll i = 0; i <= (n - 1); ++i)
    {
        pthread_mutex_destroy(&(comp[i].comp_mutex));
    }
    for (ll i = 0; i < m; i++)
    {
        pthread_join(zn[i].zone_id, NULL);
    }
    for (ll i = 0; i < m; i++)
    {
        pthread_mutex_destroy(&(zn[i].zone_mutex));
    }
    pthread_mutex_destroy(&vac_buff_mutex);
    pthread_mutex_destroy(&stud_buff_mutex);

}
```

The basic idea



The idea can be seen as zones lock the company , student ,student buff and vaccine and it locks those whenever needed in the comp_funct, student_funct and the zone_funct. Similarly , company and student can lock vaccine and student buff respectively.

void *comp_func

It creates batches, vacc per batches containing random no. of vaccines and handles all the threads of the company .Each Pharmaceutical Company manufactures its own version of a coronavirus vaccine which has the given probability of successfully administering antibodies into the student.

```
void *comp_func(void *arg)
{
    company *comp = (company *)(arg);
    flag = 0;
    while (condition)
    {
        pthread_mutex_lock(&(comp->comp_mutex));
        if (comp->batches > 0)
```

```
            pthread_mutex_unlock(&(comp->comp_mutex));
        else
        {
            ll range;
            if (flag && condition)
            {
                printf("\033[1;35mThe Vaccines by Company %Ld are finished , doing
Production .\n\033[0m", comp->num);
            }
            flag = 1;
            comp->batches = (1 + rand() % 5);
            comp->vac_per_bh = (10 + rand() % 11);
            if (condition != 0)
            {
                printf("\033[1;35mCompany %Ld has the success probability %f and is
preparing %Ld batches.\n\033[0m", comp->num, comp->prob, comp->batches);
            }
            sleep(2 + rand() % 4);
            if (condition != 0)
            {
                printf("\033[1;35mCompany %Ld has which has the success probability %f
and has prepared %Ld batches and is waiting for batches to be used and then will
resume production.\n\033[0m", comp->num, comp->prob, comp->batches);
            }
            pthread_mutex_lock(&vac_buff_mutex);
            range = comp->batches;
            for (ll i = 0; i <= (range - 1); ++i)
            {
                vac_buff[len_vac_buff] = comp;
                len_vac_buff = len_vac_buff + 1;
            }

            pthread_mutex_unlock(&vac_buff_mutex);
            pthread_mutex_unlock(&(comp->comp_mutex));
        }

        sleep(rand() % 3);
    }
}
```

void *student_func

Once the student finds a vaccination zone with an empty slot, he heads over to the vaccination zone to get vaccinated. Once the student is in the slot, he/she will signal to the zone that he has arrived. Subsequently, he/she gets vaccinated. Each Student can receive a maximum of 3 vaccinations, before the college gives up on him and he is sent home for another online semester.

```
void *student_func(void *arg)
{
    student *st = (student *)(arg);
    while (condition)
```

```
    {
        pthread_mutex_lock(&(st->stu_mutex));
ll         doses = st->tries; st->vacc = 0;

        if (0 < doses)
        {
ll              st_num = st->num; if ((st->status) != 0)
            {
                printf("\033[1;33mThe student %Ld is tested +ve for
Antibodies.\n\033[0m", st_num);
            }
            else
            {
                printf("\033[1;33mThe Student %Ld is tested -ve for
antibodies.\n\033[0m", st_num); //vac fail
            }
            if (doses > 2 || (st->status) == 1)
            {
                if ((st->status) > 2)
                {
                    printf("\033[1;33mAll the trials for the Student %Ld is exhausted
and he is returning home.\n\033[0m", st_num);
                }

                pthread_mutex_unlock(&(st->stu_mutex));
                return NULL;
            }
        }
        ll round = st->tries + 1;
        printf("\033[1;33mThe Student %Ld has arrived for the %Ld round of his
vaccination.\n\033[0m", st->num, round);

        pthread_mutex_lock(&stud_buff_mutex);

        stu_buff[len_stud_buff] = st;
        len_stud_buff = len_stud_buff + 1;
        printf("\033[1;33mStudent %Ld is waiting to be allocated a vaccination
zone.\n\033[0m", st->num); //in queue
        pthread_mutex_unlock(&stud_buff_mutex);

        while (!(st->vacc))
        {
            pthread_mutex_unlock(&(st->stu_mutex)),
                sleep(rand() % 2);
            pthread_mutex_lock(&(st->stu_mutex));
        }
        pthread_mutex_unlock(&(st->stu_mutex));
    }
}
```

void *zone_func

It waits for any company to send the batch of vaccine .Once the Vaccination zone receives a batch of coronavirus vaccine it can start vaccinating the students.

```
void *zone_func(void *arg)
{
    ll flag;
    zone *zn = (zone *)(arg);
    flag = 0;
    while (condition)
    {
        pthread_mutex_lock(&(zn->zone_mutex));
        if (zn->doses_left <= 0)
        {
            if (flag != 0)
            {
                printf("\033[1;34mThe Vaccination Zone %Ld has run out of
Vaccines.\n\033[0m", zn->num);
                pthread_mutex_lock(&(zn->comp->comp_mutex));
                zn->comp->batches = (zn->comp->batches) - 1;
                pthread_mutex_unlock(&(zn->comp->comp_mutex));
            }

            pthread_mutex_lock(&vac_buff_mutex);
            while ((condition != 0) && (len_vac_buff <= 0)) //vac buffer is empty or
not
            {

                pthread_mutex_unlock(&vac_buff_mutex),
                    sleep(rand() % 2);
                pthread_mutex_lock(&vac_buff_mutex);
            }
            if (!condition)
            {
                pthread_mutex_unlock(&vac_buff_mutex);
                return NULL;
            }
            flag = 1;
            zn->comp = vac_buff[len_vac_buff - 1];
            len_vac_buff = len_vac_buff - 1;
            ll doses_left = zn->comp->vac_per_bh;
            printf("\033[1;34mThe Company %Ld is delivering a vaccine batch to the
Vaccination Zone %Ld which the has success probability %f.\n\033[0m", zn->comp->num,
zn->num, zn->comp->prob);
            zn->doses_left = doses_left;
            printf("\033[1;34mPharmaceutical Company %Ld has delivered vaccines to
Vaccination zone %Ld, resuming vaccinations now.\n\033[0m", zn->comp->num, zn->num);

            pthread_mutex_unlock(&vac_buff_mutex);
        }
        pthread_mutex_lock(&stud_buff_mutex);
```

```
        while ((condition != 0) && len_stud_buff <= 0)
        {
            pthread_mutex_unlock(&stud_buff_mutex),
                sleep(rand() % 2);
            pthread_mutex_lock(&stud_buff_mutex);
        }
        if (condition == 0)
        {
            pthread_mutex_unlock(&stud_buff_mutex);
            return NULL;
        }
        ll less;
        if (len_stud_buff < zn->doses_left)
        {
            less = len_stud_buff;
        }
        else
        {
            less = zn->doses_left;
        }
        if (8 < less)
        {
            less = 8;
        }

        zn->slots = (1 + rand() % less);
        printf("\033[1;34mVaccination Zone %Ld is ready to vaccinate with %Ld
slots.\n\033[0m", zn->num, zn->slots);

        int count;
        student *arr[zn->slots];
        count = 0;
        for (int i = 0; i < (zn->slots); ++i)
        {
            arr[i] = stu_buff[len_stud_buff - 1];
            count = count + 1;
            len_stud_buff = len_stud_buff - 1;

            printf("\033[1;34mThe Student %Ld is assigned a slot on the Vaccination
Zone %Ld and waiting to be vaccinated.\n\033[0m", arr[i]->num, zn->num);

            if (!len_stud_buff)
                break;
        }
        pthread_mutex_unlock(&stud_buff_mutex);

        zn->doses_left = zn->doses_left - count;
        printf("\033[1;34mVaccination Zone %Ld entering Vaccination Phase.\n\033[0m",
zn->num);

        for (int i = 0; i <= (count - 1); ++i)
        {
```

```
            pthread_mutex_lock(&(arr[i]->stu_mutex));

            printf("\033[1;34mThe Student %Ld on Vaccination Zone %Ld has been
vaccinated that has success probability %f.\n\033[0m", arr[i]->num, zn->num, zn->comp-
>prob);
            arr[i]->tries = 1 + arr[i]->tries;
ll             f = vaccinated(zn->comp->prob); if (f)
            {
                arr[i]->status = 1;
            }
            arr[i]->vacc = 1,
            pthread_mutex_unlock(&(arr[i]->stu_mutex));
        }

        pthread_mutex_unlock(&(zn->zone_mutex));
        sleep(rand() % 2);
    }
}
```