```
Ayushman Panda, 2020121007 , OsN Assignment 4 Question 2
```

# Overview :

Simulating Vaccination for IIIT-Hyd campus.There are 'N' pharma companies appointed to supply vaccines .Each company has a success probability (0 - 1).There are 'V' Vaccination Zones in the campus ,ready to accept vaccines from the pharma companies.S students register to get vaccinated.

The Pharma companies produce vaccines in batches ,each batch contains random number of vaccines.The zones receive 1 batch of vaccine randomly from any of the N pharma companies,and generate slots of random size to vaccinate students,who are waiting.

After getting vaccinated,the students test for an antibody test based on the probability of the vaccine-batch they were vaccinated with.If they test positive for antibodies,they can continue their classes,else they wait to get vaccinated again.A student is vaccinated max 3 times,after which ,if he/she tests negative,is sent home for online semester.

## Running the code

```
gcc q2.c -lpthread
```

```
./a.out
```

## Input :

The Simulation Takes inputs in the following format :\ <no_of_pharma_companies(n)> \ The next n lines take the respective probabilities of n companies.

#### Example:\

5 7 20\ 0.5\ 0.6\ 0.7\ 0.7\ 0.8

## Structures for threads

#### Pharma Companies :

```
typedef struct Pharma{
    int pharma_index;
    float pharma_probab; //Success rate of vaccine
    int no_batches; //no of batches produced
    int batch_capacity; //no of vaccines in each batch
    pthread_mutex_t pharma_mutex; //mutex
    pthread_t pharma_thread; //thread
    pthread_cond_t vax_cond; //conditional variable for conditional wait
}Pharma;
```

#### Vaccination Zones :

```
typedef struct Vac_zones{
    int vax_index;
    int slot_capacity; //capacity of each slot
    int slots; //no of slots
    int occupied; //slots occupied
    int refill; //to track if its receiving the first batch or being refilled
    float batch_probab; //Success rate of current batch of vaccine
    pthread_mutex_t vax_mutex; //mutex
    pthread_t vax_thread; //thread
    pthread_cond_t stud_cond; //conditional variable for conditional wait
}Vax;
```

#### Registered Students :

```
typedef struct Students{
    int student_index;
    pthread_t student_thread;
    int round_no;
    int status; //flag : turns 1 on being tested positive for antibodies
}Students;
```

# Approach

Creating a thread for each company ,vaccination zone and student and passing to respective functions to handle them.Mutex locks and conditional waits done to control the flow of process among various functions and maintain mutual exclusivity.

```
    for(i=0;i<no_pharma;i++)
        pthread_create(&(pharma[i].pharma_thread), NULL,pharma_func,&pharma[i]);
    for(i=0;i<no_vax;i++)
        pthread_create(&(vax[i].vax_thread), NULL,vax_func,&vax[i]);
    for(i=0;i<no_students;i++)
        pthread_create(&(students[i].student_thread), NULL,student_func,&students[i]);
```

## Pharma Companies :

- Produce random number of batches X
- Sleep for a while to simulate production time
- Allocate random capacity Y for each batch
- Company ready with X batches with Y capacity each
- Does a conditional wait until all batches are distributed to vaccination zones
- Once all batches are distributed, resumes production

### Function Definition:

```
void *pharma_func(void *args)
{
    Pharma *pharma = (Pharma*)args;
    while(1)
    {
        int batches= rand()%5+1;
        printf("\x1B[1;32mPharmaceutical Company %d is preparing %d batches of vaccines which have
        success probability %f\n",pharma->pharma_index,batches,pharma->pharma_probab);
        int sleeep=rand()%4+2;
        sleep(sleeep);
        pthread_mutex_lock(&(pharma->pharma_mutex));
        pharma->no_batches=batches;
        pharma->batch_capacity=rand()%11+10;
        printf("\t\x1B[1;32mPharmaceutical Company %d has prepared %d batches of vaccines which have
        success probability %f.Waiting for all the vaccines to be used to resume production..\n",
        pharma->pharma_index,batches,pharma->pharma_probab);
        pharma_ready(pharma);
    }
}
```

```
void pharma_ready(Pharma *pharma)
{
    while(1)
    {
        if(pharma->no_batches==0)
            break;
        else
        {
            pthread_cond_wait(&(pharma->vax_cond),&(pharma->pharma_mutex));
        }
    }
```

```
    printf("\t\t\t\t\x1B[1;31mAll the vaccines prepared by Pharmaceutical Company %d are emptied.
    Resuming production now...\n",pharma->pharma_index);
    pthread_mutex_unlock(&(pharma->pharma_mutex));
}
```

## Vaccination Zone :

- Checks for available batches from either of the pharma companies
- Signals Company to get a batch
- Randomly gets a batch from a company
- Allocates random no of slots Z for vaccination
- Enters Vacciation phase with Z slots
- Does a conditional wait until all slots are occupied by students
- Allocates another Z' slots if all slots filled
- Signals for refill of vaccines with another batch of vaccines from a company,and resumes vaccination

### Function Definition :

```
void * vax_func(void *args)
{
    Vax *vax = (Vax*)args;
    while(1)
    {
        int flag = 0,i;
        for(i=0; i < no_pharma; i++)
        {
            pthread_mutex_lock(&(pharma[i].pharma_mutex));
            if(pharma[i].no_batches>0)
            {
                flag=1;
                vax->slot_capacity=pharma[i].batch_capacity;
                pharma[i].no_batches--;
                if(vax->refill==0)
                {
                    printf("\x1B[1;33mPharmaceutical Company %d is delivering a vaccine batch to Vaccination
                    Zone %d which has success probability %f\n",i+1,vax->vax_index,pharma[i].pharma_probab);
                    vax->batch_probab=pharma[i].pharma_probab;
                    vax->refill++;
                }
                else
                {
                    printf("\x1B[1;33mPharmaceutical Company %d has delivered vaccines to Vaccination zone %d,
                    which has success probability %f resuming vaccinations now..\n",i+1,vax->vax_index,pharma[i].pharma_probab);
                    vax->batch_probab=pharma[i].pharma_probab;
                }
                pthread_cond_signal(&(pharma[i].vax_cond));
                pthread_mutex_unlock(&(pharma[i].pharma_mutex));
                break;
            }
            pthread_cond_signal(&(pharma[i].vax_cond));
            pthread_mutex_unlock(&(pharma[i].pharma_mutex));
        }
        while(flag)
        {
            pthread_mutex_lock(&(vax->vax_mutex));
            if(vax->slot_capacity == 0)
            {
                printf("\t\t\t\t\t\t\t\t\x1B[1;36mVax zone %d ran out of vaccines ,
                waiting for next batch of vaccines to resume vaccination ..\n",vax->vax_index);
                pthread_mutex_unlock(&(vax->vax_mutex));
                break;
            }
            vax->slots=rand()%8+1;
            vax->occupied=0;
            if(vax->slots > vax->slot_capacity)
                vax->slots= vax->slot_capacity;
```

```
            vax->slot_capacity = vax->slot_capacity - vax->slots;
            printf("\t\t\t\t\t\t\x1B[1;35mVax Zone %d ready to vaccinate with %d slots\n",vax->vax_index, vax->slots);
            vax_ready(vax);
        }
    }
}

void vax_ready(Vax *vax)
{
    printf("\x1B[1;37mVaccination zone %d entering Vaccination Phase\n",vax->vax_index);
    while(1)
    {
        if(vax->slots == vax->occupied)
        {
            printf("\t\t\t\t\t\t\x1B[1;33mAll slots filled for vax zone %d, waiting for vacancy ...\n",vax->vax_index);
            break;
        }
        else
        {
            pthread_cond_wait(&(vax->stud_cond), &(vax->vax_mutex));
        }
    }
    pthread_mutex_unlock(&(vax->vax_mutex));
}
```

## Registered Students :

- Checks for antibodies,if found ,returned
- If antibodies not present, waits for a slot in a vaccination zone
- Signals to a vaccination zone
- Gets allocated in a random available slot in a vaccination Zone
- Sleeps for 1 sec to simulate vaccination
- Vaccinates and goes for antibody test
- If tested posotive for antibodies , status updated to 1
- Else, sent for another round of vaccination if 3 rounds of Vaccination not completed
- If tests negative even after 3rd round,sent home for online semester

### Function Definition :

```
void *student_func(void *args)
{
    Students *stud = (Students*)args;
    if(stud->status==1)
        return NULL;
    int t= rand()%10;
    sleep(t);
    stud->round_no++;
    printf("\t\t\t\t\t\t\x1B[1;36mStudent %d has arrived for his vaccination (round no :%d )\n",
    stud->student_index,stud->round_no);
    students_waiting(stud->student_index);
}
```

```
void students_waiting(int index)
{
    int flag = 0,i;
    while(!flag)
    {
        for(i=0; i < no_vax;i++)
        {
            pthread_mutex_lock(&(vax[i].vax_mutex));
            if((vax[i].slots - vax[i].occupied) > 0)
            {
                vax[i].occupied++;
                flag=1;
                printf("\t\t\x1B[1;31mStudent %d waiting to be allocated a slot  in a Vaccination zone\n",index);
                students_vaccinating(i, index);
                break;
            }

            pthread_cond_signal(&(vax[i].stud_cond));
            pthread_mutex_unlock(&(vax[i].vax_mutex));
        }
    }
}
```

```
void students_vaccinating(int i , int index)
{
    printf("\t\t\x1B[1;33mSTUDENT %d Assigned a slot at Vaccination Zone %d, waiting to be vaccinated ..\n",index, i+1);
    sleep(1);
    printf("\t\t\t\t\x1B[1;35mSTUDENT %d VACCINATED AT VAX_zone %d which had success probability %f\n",
    index, i+1,vax[i].batch_probab);
    int at=antibody_test(index,i+1);
    if(at==1)
    {
        students[index-1].status=1;
    }
    else
    {
        pthread_join(students[index-1].student_thread,0);
        pthread_create(&(students[index-1].student_thread), NULL,student_func,&students[index-1]);
        //student_func(&students[index-1]);
    }
    pthread_cond_signal(&(vax[i].stud_cond));
    pthread_mutex_unlock(&(vax[i].vax_mutex));

}
```

### Antibody test Function :

```
int antibody_test(int index,int zone_index)
{
    printf("\t\t\t\t\t\x1B[1;37mStudent %d going for antibody test :\n",index);
    int i = index-1;
    float r = (rand()/(double)RAND_MAX);
    //float r= 1.0;
    //printf("\n\nvalue of r :%f , zone no : %d zone batch probab:%f\n\n",r,zone_index,vax[zone_index].batch_probab);
    if(students[i].round_no==3)
    {
        if(r>vax[zone_index].batch_probab)
            printf("\t\t\t\t\t\x1B[1;32mStudent %d Tested Positive for antibodies\n",index);
        else
        {
            printf("\t\t\t\t\t\x1B[1;31mStudent %d Tested Negative for antibodies\n",index);
            printf("\x1B[1;32m3 trials of Vaccination complete, Student %d sent home for online semester,
            Take care!\n",index);
        }
        return 1;
    }
    if(r>vax[zone_index].batch_probab)
    {
        printf("\t\t\t\t\t\x1B[1;32mStudent %d Tested Positive for antibodies\n",index);
        return 1;
    }
    else
    {
        printf("\t\t\t\t\t\x1B[1;31mStudent %d tested negative for antibodies,will try again .. \n",index);
        return 0;
    }

}
```

## NOTE ::

Colours and Space Indentation added for better understanding of the simulation\ Sleep statements used at appropriate places to simulate production,vaccination ,etc to make the simulation more realistic