Back To College

Company threads produce random number of batches with randon number of vaccines and distribute each batch randomly to Vaccination Zone threads which provides slots to students and each vaccination has a success probability.

Running The Program

- Run gcc q2.c -lpthread;
- Run ./a.out

Input Format

- Enter the no. of Companies, Vaccination Zones, Students.
- Then, Enter Probability of success rate of each Company.

Functions

randomno

```
ll num;
num = (rand()%(h-l+1))+l;
return num;
```

Produces Random number between l, h.

randomprob

```
ld r = (ld)rand() / RAND_MAX;
return r > prob;
```

Generates 0 or 1 based given probability.

phar_company

Each Company produces vaccines and resume production only when vaccines of all the batches are used. It also checks for when no. of students left <= 0.

```
    ll w = randomno(2,5);
    ll r = randomno(1,5);
    ll p = randomno(10,20);
    comp->vaccines = p;
    comp->batches_num = r;
    printf("\033[0;33mPharmaceutical Company %lld is preparing %lld batches of
Vaccines which have success probability %Lf\n",comp->id,r,prob[comp->id-1]);
    printf("\033[0m");
    sleep(w);
    printf("\033[0;33mPharmaceutical Company %lld has prepared %lld batches of
Vaccines which have success probability %Lf\n",comp->id,r,prob[comp->id-1]);
    printf("\033[0m");
```

This part of function prouduces r batches of p vaccines waiting for w secs to produce.

```
    while(1)
    {
        ll sum=0;
        pthread_cond_wait(&(comp->cond),&(comp->mutex));
        for(ll i=0;i<m;i++)
        {
            if(Zone[i].company==comp->id)
            {
                sum+=Zone[i].vaccines;
            }
        }
        if(sum==0)
        {
             printf("\033[0;31mAll the vaccines prepared by Pharmaceutical Company
%lld are emptied. Resuming Production now\n", comp->id);
             pthread_cond_signal(&(comp->cond));
             break;
        }
    }
```

Company conditionally wait (to avoid busy waiting) until all the vaccines of all the batches produced by that company are used. Resume production by signalling wait and resuming vaccination.

vacc_zone

Each Vaccination Zone checks for already prepared batches and assign random number of slots to Vaccination Zones.

```
 if(Company[i].batches_num==0)
{
    // printf("\033[0;31mAll the vaccines prepared by Pharmaceutical Company %lld are
emptied. Resuming Production now\n", Company[i].id);
    pthread_cond_signal(&(Company[i].cond));
}
```

It signals Company to check if all the Vaccines are used or not as this particular Company has given all it's prepared batched to Vaccination Zones.

```
 printf("\033[0;36mPharmaceutical Company %lld is delivering vaccine batch to
Vaccination Zone %lld which has success probability %Lf\n",Company[i].id,vz-
>id,prob[i]);
printf("\033[0m");
Company[i].batches_num--;
vz->vaccines = Company[i].vaccines;
vz->probability = prob[i];
vz->company = Company[i].id;
sleep(2);
printf("\033[0;36mPharmaceutical Company %lld has delivered vaccines to Vaccination
Zone %lld, resuming vaccinations now\n",Company[i].id,vz->id);
printf("\033[0m");
```

Vaccization Zone vz takes batch from Company if there are any currently present.

```
vz->free = 1;
ll given = 8;
vz->slot = min(min(given,vz->vaccines),o);
```

Slots are provided to Vaccination Zone.

```
if(vz->vaccines<=0)
{
    printf("\033[0;32mVaccination Zone %lld has ran out of vaccines\n", vz->id);
    printf("\033[0m");
    pthread_mutex_unlock(&(vz->mutex));
    continue;
}
```

If Vaccination Zone is emptied out of Vaccines, then it asks for new batch from the Company.

student_slot

This Function allow the arrival of each student and allots a slot to each of them. Then, checks that the particular student tests postive or negative for anti bodies. If, it test positive, it is allowed to stay and if it tests negative for 3 times, it is send home.

```
time = randomno(1,10);
printf("\033[0;35mStudent %lld has arrived for %lld round of Vaccination\n", sd->id,
sd->round);
printf("\033[0m");
sleep(time);
printf("\033[0;35mStudent %lld is waiting to be allocated a slot on a Vaccination
Zone\n", sd->id);
printf("\033[0m");
```

Random arrival of each student.

```
Zone[i].slot--;
Zone[i].vaccines--;
printf("\033[0;35mStudent %lld assigned a slot on the Vaccination Zone %lld and
waiting to be Vaccinated\n", sd->id, Zone[i].id);
printf("\033[0m");
sleep(1);
printf("\033[0;35mStudent %lld on Vaccination Zone %lld has been vaccinated which has
a success probability %Lf\n", sd->id, Zone[i].id, Zone[i].probability);
printf("\033[0m");
sd->probability = Zone[i].probability;
```

Assign a slot to a Student.

```
sd->test = randomprob(sd->probability);
if(sd->test==1)
{
        o--;
        printf("\033[0;35mStudent %lld tested positive for antibodies\n", sd->id);
```

```
        printf("\033[0m");
        pthread_mutex_unlock(&(sd->mutex));
        break;
}
else
{
    if(sd->round==3)
    {
        printf("\033[0;31mStudent %lld tested negative for antibodies 3 times, now
it's time to go home :(\n"sd->id);
        printf("\033[0m");
        pthread_mutex_unlock(&(sd->mutex));
        break;
    }
    sd->round++;
    printf("\033[0;35mStudent %lld tested negative for antibodies\n", sd->id);
    printf("\033[0m");
}
```

Randomprob function gives 0 or 1 to each student based on probability given and if it tested positive(getting values as 1) while be out of the loop. Otherwise, it will get vaccinated for maximum 3 times.

Main

- Take input
- Initialise threads and joins them.
- Initialise Mutexes.
- Destroys Mutexes.