Explaination Of Logic:

Note that company, student , vaccination zone are threads and each will be passed to their funtions company , student , vacination given is the detailed explaination as to what each thread will do in their respective functions. Basic overview would be to treat them as threads and will execute together parallely. Student thread will try to get the vaccines from the zones but the locks no two students will use the same vaccine. The code is nicely commented please go through it incase of any doubt. Also all the variable names are easy to understand.

Company:

Each company will have to check if the people who already got result to be less than total students then the company will check if it should produce. But there might be the case that even though the everybody is done with their result if the company already entered the by checking the condition which is holding good a while ago then the execution will not stop.

Basically the production will start immediately after the completion of the batches it produced so there is no relation with the student result but as soon as the students are done then the production will also stop after atmost one more production cycle this is because at the time of checking there may be few students who are yet to get result I didnt keep lock because just to make it happen there will considerable loss of delay so as a design decision I have decided to do this way.

```
if(batchescompleted[num]==batchesproduced[num]) // if produced = completely
exhauseted batches then reproduction needs to be done.
{
    printf("All the vaccines prepared by Pharmaceutical Company %d are
emptied. Resuming production now.\n",num);
    int ti=randomnumber(2,5); // generting random number time for which it
should wait.
    int vvv=randomnumber(1,5);// generting random number of batches
    printf("Pharmaceutical Company %d is preparing %d batches of vaccines
which have success probability %Lf\n",num,vvv,probability[num]);

    sleep(ti);
    pthread_mutex_lock(&companylock[num]);

    batchesproduced[num]=vvv; // Once the sleep is done the number of
batches produces is updates.
    batchesuntouched[num]=batchesproduced[num];// number of batches which
are not even touched are initalized to number of batches produced.
    vacinesinbatch[num]=randomnumber(10,20);// This will generate vacines
per batch
    batchescompleted[num]=0;// number of vaccines completd is 0.
    printf("Pharmaceutical Company %d has prepared %d batches of vaccines
which have success probability %Lf. Waiting for all the vaccines to be used
to resume production\n",num,batchesproduced[num],probability[num]);
    pthread_mutex_unlock(&companylock[num]);
}
```

Once we find that the batches need to produce we produce random number b/w 1 to 5 batches and random number of vaccines for some random amount of time. Once we are done with this execution we will update the batches produced by this comapny and also we will update the batches which are completely done and the batches which are completely untouched. Batches produced as the name suggests the number of batches produced. And the batches completed will be 0 since at the time of initialization there cannot be any batches which are already done.Batches untouched is the array which we will refer when should production begin If there is atlest one untouched batch in the current company then we will not start production. This untouched count will decrease when a zone which took batch from this company got exhausted.

Vaccination:

In this we will first check if there are anymore students who need result. If there are any then we will check the number of vaccines left in the current zone. If there are none available then we need to go to the company to get a batch of vaccines. So untill we have zero vaccines we will keep on checking companys if anyone of them has untouched batches so that we can take one of them and give it to the current zone. After we see any company having untouched bathces we will give that batch to the exhausted zone. Along with that we will update number of vaccines in it and also the probability of the comapny's as the probability of the zones vaccine. These updations are done in mutex locks so that we will not get inaccurate result.

```
pthread_mutex_lock(&companylock[j]);
if(batchesuntouched[j]>0) // we are checking if ith company has any
untouched vaccines.
{
    currcomp=j; // if it has untouched batch the give it to this zone and
update the currcomp(any) variable.
    printf("Pharmaceutical Company %d is delivering a vaccine batch to
Vaccination Zone %d which has success  probability
%Lf\n",j,num,probability[j]);
    sleep(1);
    sleep(1);
    printf("Pharmaceutical Company %d has delivered vaccines to Vaccination
zone %d, resuming vaccinations now\n",j,input->id);
    prob[num]=probability[j]; // Make the probability of the current zones
vaccines equal to the company from which the vaccines are taken.
    batchesuntouched[j]--; // Number of untouched batches need to be
decresed.
    pthread_mutex_lock(&vacinelock[num]);
    vacines[input->id]=vacinesinbatch[j]; // number of vacines are made
equal to the number of vaccines in the company's batch it took from.
    pthread_mutex_unlock(&vacinelock[num]);
}
    pthread_mutex_unlock(&companylock[j]);
```

Once that is done we will find the slots of the zone using the conditions mentioned, we will store this as the slotsleft for this current zone so that students can join until there are slots in the current zone or there are no students who are waiting, but since we need to stop the vaccine zone process I used while loop to stop until the following conditions are not satisfied, Immediately after breaking the loop we will set the slotsleft

in the current zone to 0 so that no new students can join. By doing this we ensured that our zone will wait until everybody who needs to join are joined. We will also store the students entered in the loop using peopleleft[i] to be the number of students joined in ith zone.

```
int vv=min(8,min(vacines[num],numberofstudentwaiting)); //This is to
calculate the maximum number of slots which the zone can allocate.
if(vv==0)continue;
pthread_mutex_lock(&vacinelock[num]);
peopleleft[num]=0; // This is the variable which stores number of students
in the current zone.
pthread_mutex_unlock(&vacinelock[num]);
printf("Vaccination Zone %d is ready to vaccinate with %d slots\n",num,vv);
pthread_mutex_lock(&vacinelock[num]);
slotsleft[num]=vv; // Slots left is made equal to the number we cacluted
before.
pthread_mutex_unlock(&vacinelock[num]);
while(slotsleft[num]>0&&numberofstudentwaiting>0&&peoplewhogotresult<number
student){
    // printf("%d %d===\n",peoplewhogotresult,numberofstudentwaiting);
    // This statement makes sure that all the possible stiudets who are
wating will be given slots if they are availabel.
}
pthread_mutex_lock(&vacinelock[num]);
slotsleft[num]=0; // We will now stop the intake of the students by making
the slots left in  zone equl to 0, irrespectiuve of what happnes latter.
pthread_mutex_unlock(&vacinelock[num]);
```

Now if still peopleleft[i] is 0 then there are no students who joined so we go back to initial state, because its just waste of time executing when no one is there if there are some studenst then we will enter the zone and then we will set inzone[i] to 1 after we complete the exectution of the process of vaccinating the student which will take some 2sec, it indiacates that the zone is currently allowing the students to reveal their result in the executing zone, if it is 0 then it is not allowing anymore in the zone. After converting to inzone[i] to 1 we will check that everyone who joined will exit this can be achieved by the peopleleft[i] we incremented before. inzone vaiable gives us the flexibility as to choose when to show the result of the student without this we may create some wring ordering of statements.

```
sleep(3); // processing time
pthread_mutex_lock(&vacinelock[num]);
inzone[num]=1; // This variable represents that all the processes in the
current zone are free tprint their result.
pthread_mutex_unlock(&vacinelock[num]);
while(peopleleft[num]>0&&peoplewhogotresult<numberstudent){} // makes sure
that all the students in the curtrent thread gave the result.
pthread_mutex_lock(&vacinelock[num]);
inzone[num]=0; // Once all are done we should once again lock this zone.
pthread_mutex_unlock(&vacinelock[num]);
printf("Vaccination Zone %d finished Vaccination Phase\n",num);
```

Student:

In the student thread we will ask it to wait for some random time so as to randomize already randomized arrival time, and we will increase the threads which are waiting. We do these things in mutex lock so that the number of students waiting will be updated properly.

```
sleep(randomnumber(1,3));
studentinfo* input = (studentinfo*)inp;
pthread_mutex_lock(&lock);
numberofstudentwaiting++;   // This variable represents the number of
students who did not get any result yet this variable is to keep track of
number of students who should still be processes.
pthread_mutex_unlock(&lock);
```

After that we will exit only if the student either tested positive or have completed 3 rounds testing negative in all the cases. Then we loop through the vaccination zones to check if there are any vaccination zones which have slots left in them. We will also increment the peopleleft so that the we can use it in the vacination function by the vaccination zone. We will also decrease the vacines left in the vaccination zone so that it will now have the correct number of vaccines, even this will dones in lock so that no two student will be able to take the same slot nor will be able to decrease the same count.

```
pthread_mutex_lock(&vacinelock[i]);
if(slotsleft[i]>0) // Slotsleft in the vaccination zone i.
{
    pthread_mutex_lock(&lock);
    numberofstudentwaiting--; // Since the slots are there numberof
students ating are reduced since this student is assigned to a vaccine
zone, and rest of the vaccine zones shouldnt confuse hence reducing the
thread.
    pthread_mutex_unlock(&lock);
    peopleleft[i]++;    // This keeps track of the people in the current
zone.
    slotsleft[i]--; // Decreased the slots left in the zone.
    vacines[i]--;   // After this student came vaccines left in  the zone
should be reduced.
    flag=-1;        // this is used in two ways one to print next round if
failed or to update the variables after this if condition.
    printf("Student %d assigned a slot on the Vaccination Zone %d and
waiting to be vaccinated\n",input->id,i);
}
pthread_mutex_unlock(&vacinelock[i]);
```

Like we waited in the vaccination zone for everone to wait here also we will wait for all other potential students to join then until then it would just loop over and over. Then we wait for exectution to be done until then we will wait this can be done using inzone[i] to be set to 1. Then we use the random numebrs to predict the result of the person who took the test.

```
roundnumber[input->id]++; //
while(slotsleft[i]>0&&numberofstudentwaiting>0&&peoplewhogotresult<numberst
udent){} // To make sure that this thread is waiting untill all the
required threads board in the vaccination zone.
while(inzone[i]==0){}// when ith vaccination zone is being near completion
of execution then inzone[i] will be turned to one so that the processes in
the zone can be now print their messages..
printf("Student %d on Vaccination Zone %d has been vaccinated which has
success probability %Lf\n",input->id,i,prob[i]);
ll var=100*(prob[i]); // scaling the current probability by 100
```

Depending on whether he tested poitive or negative we deicde what to do if it was positive then we ask the student print the same and will no longer be in race of getting zone. We decrese the peopleleft we discusse previously so that we will know as to for how much more time the zone thread need to wait until all the threads which joined it are terminated.

```
// this is to check if the randomly generated number is less thancalculated
thing or not.
if(var>=randomnumber(0,100))
{
    printf("Student %d has tested positive for antibodies.\n",input->id);
    ishecured[input->id]=1;
    pthread_mutex_lock(&lock);
    peoplewhogotresult++; // if any student tested positive then he is
immediuately sent ot college. So the people who got result got incremented.
    pthread_mutex_unlock(&lock);
    pthread_mutex_lock(&vacinelock[i]);
    peopleleft[i]--; // This varaible will take care as to all persons who
entered the zone exited or not. This is incremented when somenbody enters
and decrements when someone leaves.
    pthread_mutex_unlock(&vacinelock[i]);
    return NULL;
}
```

If it is negative then we need to increase the number of students waiting so that this thread can take a place in a zone if there are any vacancies. In this case also we decrese the peopleleft[i] so that the zone will be able to track if there are any more students to tell their result. Ofcourse if the number of rounds are greate than 3 then we will no longer try on this student so we exit by saying the same.

```
printf("Student %d has tested negative for antibodies.\n",input->id);
pthread_mutex_lock(&lock);
numberofstudentwaiting++; // If the person tested negetive then number of
threads waiting should be increased because
pthread_mutex_unlock(&lock);
pthread_mutex_lock(&vacinelock[i]);
peopleleft[i]--; // This varaible will take care as to all persons who
entered the zone exited or not. This is incremented when somenbody enters
and decrements when someone leaves.
```

```
pthread_mutex_unlock(&vacinelock[i]);
break;
```

Assumptions:

Since there are no vaccines in any zone initially they will say that they dont have any right at the beginning of the code execution they will print that they are exhausted.

Maximum number of student/comanies/vaccines are no more than 1000.