Detailed Explaination Of Logic:

To start with I created a threads for each performer which will go to fperform, in this function each of the performer will wait untill the time of arrival comes. Also all the variable names are easy to understand.

```
    for(int i=1;i<=k;i++)
    {
        pthread_create(&extrathread[i], NULL,fperform,
(void*)performerinput1[i]);
    }
```

After the performer completes his respective waiting time he will try to go to every possible stage, ie he will try get acoustic stage, electric stage and even a stage where there is already a performer performing. We will deal with individually later, but the basic idea is which ever it gets first be it be electric or acoustic it will occupy it, and will make sure that rest of them will not be occupied by the same performer. As far as implementation is concerend I have created three threads each will go to their respective function where they will try to get into the stage. I waited all the threads to come and then I exited from this function.

```
    pthread_create(&performer1[num],NULL,acoustic,
(void*)performerinput1[num]);
    pthread_create(&performer2[num],NULL,electric,
(void*)performerinput2[num]);
    pthread_create(&performer3[num],NULL,both,(void*)performerinput3[num]);
    pthread_join(performer1[num],NULL);
    pthread_join(performer2[num],NULL);
    pthread_join(performer3[num],NULL);
```

Coming to how I have checked whether there is a stage of particular kind I did the follwing for each kind of stage.

Acoustic:

Incase of acoustic an instrument Bass is not allowed so I have checked if current performer is using bass if he is using bass he has no chance of performing on acoustic stage, For maintaining whether a attempt made by a performer is sucessful or not I made an array didheperform.

```
int didheperform[405][3]={0};
```

Intialized to zero, if there is a stage where it failed to get the stage for some reason then didheperform[performernumber][0] is made -1, if the stage is acoustic. When the current attempt of aquiring the stage failed then I will check if rest all the attempts made by this performer also failed, if every other attempt also failed then there is no point in waiting then I will print that the performer didnt find any

stage. There may be a case where two or more threads may be acessing the same variable didheperform so I made mutex lock so that no other process will be able to interfere.

```
pthread_mutex_lock(&performerlock[num]);
    if(instrument[num]=='b')
    {
        didheperform[num][0]=-1; // this is saying that whosoever the
perfoemer maybe his attempt to perform on acoustic stage failed.
        if(didheperform[num][1]+didheperform[num][2]==-2) // If the other
two already failed then the performer can leave the college with no further
delay.
        {
            printf("\033[1m%s %c left because of
impatience\n\033[0m",name[num],instrument[num]);
            pthread_mutex_unlock(&performerlock[num]);
            return NULL;
        }
        pthread_mutex_unlock(&performerlock[num]);
        return NULL;
    }
    pthread_mutex_unlock(&performerlock[num]);
```

I created a semaphore semacoustic which is initialized to number of acoustic stages, this will make sure that no more than 'a' performers are there in a stage (There may be singers joining them later), the thread will try to into the critical section but it is protected by the semacoustic semaphore where it should wait for no more than t seconds. For this I need absolute time since Epoch 1970-01-01 00:00:00 +0000 (UTC) For getting the current time from that point of time I used clock_gettime in which it will populate the struct timespec I passed it, On return the struct will have tv_sec which has the time from epoch, so for getting the time untill which this performer will try for this thread would be t sec _ absolute time until now.

```
struct timespec ts;
clock_gettime(CLOCK_REALTIME, &ts);
ts.tv_sec += t;
```

Once I have the time untill which it can run I will use sem_timedwait funtion to get into the critical section if I can else I will check its return status if it is -1 then within the time slot t it was not bale to get the critical section it was looking for. As described earlier since the performers attempt to aquire this stage failed he note it and will see of others also failed, if they aslo failed then he will say that he is leaving since there is no stage where he was able to enter.

```
if(sem_timedwait(&semacoustic,&ts)==-1) // This will return -1 when the
time slice expired.
{
    pthread_mutex_lock(&performerlock[num]);
    didheperform[num][0]=-1; // As earlier this meant that he failed to
aquire stage within time.
```

```
    if(didheperform[num][1]+didheperform[num][2]==-2) // if this the last
to fail then the performercan leave the college without performing.
    {
        printf("\033[1m%s %c left because of
impatience\n\033[0m",name[num],instrument[num]);
        pthread_mutex_unlock(&performerlock[num]);
        return NULL;
    }
    pthread_mutex_unlock(&performerlock[num]);
    return NULL;
}
```

If it sucessfully gets into the critical section within that time limit then also it should check if performer already aquired any other stage in which case this should exit immediately because no performer can perform on two stages at one instance, I used didheperform[performernumber][stagetype] to know if there is a stage which he already got, if he aquired a stage type then that particular thing is made equal to 1 if it failed we will make it equal to -1. So depending n whether some other stage is taken by this performer or not we will continue further or we will return since if something is already taken then trying to do something on the other stage doesnt make sense. Also we need to make sure that the same performer is not updating the variable at same time hence I locked this section of code.

```
pthread_mutex_lock(&performerlock[num]);
didheperform[num][0]=1; // Incase he gets the stage within the time then he
will be given thstage.
if(didheperform[num][1]==1||didheperform[num][2]==1) // he should check if
he already is sucessfuin getting other stages inthat case since he already
got those he can no longer required to getheis stage hence he will leave
this stage immediately.
{
    pthread_mutex_unlock(&performerlock[num]);
    sem_post(&semacoustic);
    return NULL;
}
pthread_mutex_unlock(&performerlock[num]);
```

Once I am sure that this is the first stage it gets then now I need to know on which stage is he going to play, untill now the only thing we know it that there is some acoustic stage which is free, but we dont know the stage number. For finding the stage number we loop over all the available acoustic stages and will find a stage which is not yet been occupied. Once I see such a stage I will lock everything now and will say that a performer who entered this critical section took this stage so that rest cannot take this stge, lock will ensure that no two performers will take the same stage. I maintained an array isstagefree[i] which will say the status of the stage i.

If isstagefree[i]=0 then no one is there on the stage. If isstagefree[i]=1 there is singer who is performing. If isstagefree[i]=2 there is performer other that singer performing. If isstagefree[i]=3 then there is a singer who joined a performer (not singer) Based on the performers instrument I updated this array.

```
for(int i=1;i<=a;i++)
{
    // iterating through all the acoustic stages available to see if
anyoine is free.
    pthread_mutex_lock(&stagelock[i]);
    if(isstagefree[i]==0)
    {
        if(instrument[num]=='s')
        isstagefree[i]=1;// depending on whether he is singer or not we
will update the personwho are currently performing on this stage.
        else isstagefree[i]=2;
         flfl=1;
    }
    pthread_mutex_unlock(&stagelock[i]);
    if(flfl==1)
    {
        sta=i; // storing the stage number.
      pthread_mutex_lock(&stagelock[i]);
        performeronstage[i]=num; // Storing the performer index at this
stage number.
        pthread_mutex_unlock(&stagelock[i]);
        break;
    }
}
```

I created an array istheresinger[num] which says if there is a singer who joined with a performer. I will make it 0 so that initially there is no one along with the performer. I the current performer is not a singer then there is possibility that a singer may join him while he is performing so I createn a semaphore commom which stores the number of singers who are currently permitted to join a performer, intially it is made 0, but if some performer other that singer got the stage he will increment it saying that there can be singer who is allowed to come and join him.

```
pthread_mutex_lock(&performerlock[num]);
istheresinger[num]=0; // Making sure that we initialize that there is no
additional singer alonwith our current performer.
pthread_mutex_unlock(&performerlock[num])
if(instrument[num]!='s')
sem_post(&common);
```

Now we will sleep for some random time between t1 to t2, then if we are not singer there can be someone who joined us so we need to check the staus of the stage if isstegfree[i]=3 then there is someone who joined then we will wait fro some moer time. But consider a case where a singer joined and just before making isstagefree[i]=3 that thread stopped then we would not take this case. To avoid this case we did sem_trywait on that common semaphore so that is it failed then there should be some singer who joined him so in this case also we extend the performance time by 2sec.

```
if(instrument[num]!='s')
{
   int flag=-1;
   if(isstagefree[sta]==3)
   {
       sleep(2);
       flag=1;
   }
   else if(sem_trywait(&common)==-1)
   {
       sleep(2);
       flag=1;
   }
}
```

After this we reset all the array values so that this stage is available and there is no one on this stage who are performing.

```
pthread_mutex_lock(&stagelock[sta]);
isstagefree[sta]=0; // Making this stage free for others to occupy.
performeronstage[sta]=0;// No one is there on the current stage.
pthread_mutex_unlock(&stagelock[sta]);
```

Once the execution is done we will create a thread for this performer and pass it to coordinator function where it will wait to get T-shirt. If applicabe ie if there is a singer who joined him in the middle of the performence we will create a thread for him as well.

```
pthread_create(&sparethreads[num],NULL,coordinator,(void*)spareinput[num]);
if(sinind>0) // If there is a singer additionally joining  in the middle
then even he must go andcollect T-shirt.
{
    pthread_create(&sparethreads[sinind],NULL,coordinator,
(void*)spareinput[sinind]);
    pthread_join(sparethreads[sinind],NULL);
}
pthread_join(sparethreads[num],NULL);
```

Electrical stage:

I think based on my above explaination understanding this should be cake walk, So I will only explain the cases where there is a difference between this and acoustic stage. I will use semelectric seaphore which is initialized to number of electricv stages instead of semacoustic stages. And will iteralte throuigh stages from a+1 to e for electric stages constrast to 1 to e in acoustic and importantly I will check If the is voilin instead of bass. Other than that the code and logic is exactly same please refer to above explaination for the same.

Both:

This is the function which the threads go to aquire a place with a performer who is exectuing in any of the stages, as given in the question onlysinger will be able to enter a performer so the rest of the performers should not be allowed to quire a stage through this mechanism, so firstly I will check if the performer is singer else I will mark that this possiblity of aquiring is not valid and will check if other threads of the performer also exited if yes then I will print the message that this performer can no longer perform.

```
if(instrument[num]!='s')
{
    // since only singer can join a musican in middle rest all are made as
unsucessful.
    didheperform[num][2]=-1;
    if(didheperform[num][0]+didheperform[num][1]==-2)// if rest all are
already unsucessfull thenstraight away say that this performer has no
option other than to leave.
    {
        printf("\033[1m%s %c left because of
impatience\n\033[0m",name[num],instrument[num]);
        pthread_mutex_unlock(&performerlock[num]);
        return NULL;
    }
    pthread_mutex_unlock(&performerlock[num]);
    return NULL;
}
```

Then I will do timed wait and will enter the critical section simialr to other stages I will check if this is the last stage to fail of first to succeed in either of the case I will print the messages corresponingly in case of latter we will move ahead for finding the stages where the singer is going to join. When I see a stage which is occupied by a performer whoi is no singer I will update its status so that the performer joined it and will print the message accordingly.

```
for(int i=1;i<=a+e;i++)
{
    pthread_mutex_lock(&stagelock[i]);
    if(isstagefree[i]==2)// if there is some performer other than singer
who is currentlexecuting.
    {
        isstagefree[i]=3; // we will update it to singer jopined him in the
middle.
        pthread_mutex_unlock(&stagelock[i]);
        istheresinger[performeronstage[i]]=num; // We will also make an
updation that there is aadditional singer who is singing.
        printf("\033[1;36m%s joined %s's performance, performence extended
by 2sec\n\033[0m",na[num],name[i]);
        break;
    }
    pthread_mutex_unlock(&stagelock[i]);
}
```

In this way I will acieve the given tak without busy waiting.

Assumptions made:

Number of stages, performers (including singers), coordinators should be less than 400.