Musical Mayhem

The college is opening for the new semester, and the campus is overrun with excited first years and their parents. The Music Club has decided to host an event on the first day to entertain all the newcomers.

Run the code

- **Complilation**

```
$ gcc q3.c -lpthread
```

- **Execution**

```
$ ./a.out
```

Input

Enter the values k, a, e, c, t1, t2, t as mentioned in the question. It will be followed by k lines, each line containing name of the musician, his instrument (p, g, v, b, s), and time of arrival.

```
4 1 1 2 10 5
Tanvi p 0
Sudh b 1
Manas g 0
Sriya v 1
Pahun s 1
```

Example input

Explaination of code

Variables and arrays

k - total number of musicians a - number of acooustic stages e - number of electric stages c - number of coordinators t1 - minimum performance time t2 - maximum performance time astc - number of unoccupied acoustic stages elec - number of unoccupied electric stages astc_singer - number of acoustic stages where singer is not performing elec_singer - number of electric stages where singer is not performing acoustic_m_name[1007][50] - stores the name of the musician performing on each acoustic stage electric_m_name[1007][50] - stores the name of the musician performing on each electric stage acoustic_m_inst[1007][50] - stores the instrument of the musician performing on each acoustic stage electric_m_inst[1007][50] - stores the instrument of the musician performing on each electric stage

Semaphores

acoustic - Allows only maximum a musicians to perform on acoustic stages electric - Allows only maximum e musicians to perform on electric stages stage - Allows only maximum a+e musicians to perform at a time singers - Allows only maximum a+e singers to perform at a time tshirt - Allows at max c musicians to collect tshirt at a time

Mutex

acoustic_stage - synchronises all the threads that are claiming for an acoustic stage, so that multiple musicians do not claim a single stage at a time electric_stage - synchronises all the threads that are claiming for an electric stage, so that multiple musicians do not claim a single stage at a time as[1007] - synchronises threads on individual acoustic stage, also helps in signaling singer who is performing with a musician when the performance gets over on an acoustic stage es[1007] - synchronises threads on individual electric stage, also helps in signaling singer who is performing with a musician when the performance gets over on an electric stage

Structures

```
struct musician{
    char name[50]; \\name of musician
    char inst[50]; \\instrument he is performing
    int time; \\time of his arrival at Srujana
};
```

Basic Logic

A seperate thread is created for each musician and execute different functions according to their instruments. Violinist threads execute only_a() function which allots only acoustic stages, bassist threads execute only_e() function which allots only electric stage. Pianists and guitarists execute both() function which allots any of the acoustic or electric stage that is available for the performace. Singers are dispatched to execute only_singers() function which allots any available stage for a singer. Musician arrives at Srujana as per the time given during input. He waits till any stage is allocated. If his waiting time exceeds t, he leaves Srujana. Next, we loop over all the stages to check which stage can be alloted. The musician performs on that stage for some time between t1 and t2 and then checks if some singer had joined him during his performance. If yes, he further waits for 2 sec and then leaves the stage. He also unlocks as[l]/es[l] which signals the singer performing with him (if any) to leave the stage. Further he signals all the semaphores for which he had waited. In case of singers, the stage which has no singer gets alloted. If the singer is performing with some musician, when the as[l]/es[l] gets unlocked, singer will leave the stage. The counters are also incremented / decremented before and after the performance. Then the musicians collect tshirt from the coordinator.

```
if(strcmp(inst[i],"p")==0 || strcmp(inst[i],"g")==0){
    pthread_create(&musicians[i], NULL, both, (void*)(m));
}
else if(strcmp(inst[i],"v")==0){
    pthread_create(&musicians[i], NULL, only_a, (void*)(m));
}
else if(strcmp(inst[i],"b")==0){
    pthread_create(&musicians[i], NULL, only_e, (void*)(m));
}
else if(strcmp(inst[i],"s")==0){
    pthread_create(&musicians[i], NULL, only_singers, (void*)(m));
}
```

void only_a()

Musician arrives at Srujana as per the time given during input. He waits till any acoustic stage is allocated. If his waiting time exceeds t, he leaves Srujana.

```
struct timespec ts;
struct musician *inputs = (struct musician*)inp;
sleep(inputs->time);
printf("\033[1;37m%s %s arrived\n\n\033[0m",inputs->name,inputs->inst);
clock_gettime(CLOCK_REALTIME, &ts);
ts.tv_sec += t;
int x = sem_timedwait(&acoustic, &ts);//wait till an acoustic stage is allocated or
waiting time exceeds t
if(x==-1){// if the waiting time exceeds t, then leave srujana
    printf("\033[1;32m%s %s left because of impatience\n\n\033[0m",inputs->name,
inputs->inst);
}
```

Next, we loop over all the acoustic stages to check which stage is free. The musician performs on that stage for some time between t1 and t2 and then checks if some singer had joined him during his performance. If yes, he further waits for 2 sec and then leaves the stage. He also unlocks as[l] which signals the singer performing with him (if any) to leave the stage. Further he signals all the semaphores for which he had waited.

```
sem_wait(&stage);//if an acoustic stage is allocated, we also need to increment the
stage semaphore
int l;
int k = random_func(t1,t2);
pthread_mutex_lock(&acoustic_stage);//to ensure that two musicians do not claim the
same stage simultaneously
//check which stage is free
for(int i=1; i<=a; i++){
    if(a_stage[i]==0){//check if any performer is performing of the stage
        a_stage[i]++;//increment the individual stage performer counter
        pthread_mutex_unlock(&acoustic_stage);//unlock the acoustic stage mutex
        l = i;
        // printf("%d %d\n\n", l, i);
        strcpy(acoustic_m_name[l], inputs->name);
        strcpy(acoustic_m_inst[l], inputs->inst);
        astc--;//decrement the number acoustic stages available
        break;
    }
}
printf("\033[1;35m%s performing %s at acoustic stage %d for %d sec\n\n\033[0m",
inputs->name, inputs->inst, l, k);
pthread_mutex_lock(&as[l]);
sleep(k);
if(a_stage[l]==2){ //if a singer joins the musician during his performance wait for 2
sec
    sleep(2);
}
pthread_mutex_unlock(&as[l]);
printf("\033[1;34m%s's performance at acoustic stage %d finished\n\n\033[0m",inputs-
```

```
>name, l);
sem_post(&acoustic);//acoustic stage deallocated
sem_post(&stage);//if an acoustic stage is deallocated, we also need to decrement the
stage semaphore
astc++;//increment the number acoustic stages available
a_stage[l]=0;//decrement the individual stage performer counter
```

Next he collects tshirt from the coordinator

```
sem_wait(&tshirt);//wait for coordinator to be free to get the tshirt
printf("\033[1;36m%s collecting t-shirt\n\n\033[0m",inputs->name);
sleep(2);
sem_post(&tshirt);//leave the cooridnator so that he can attend other musicians
```

void only_e()

Musician arrives at Srujana as per the time given during input. He waits till any electric stage is allocated. If his waiting time exceeds t, he leaves Srujana.

```
struct timespec ts;
struct musician *inputs = (struct musician*)inp;
sleep(inputs->time);
printf("\033[1;37m%s %s arrived\n\n\033[0m",inputs->name,inputs->inst);
clock_gettime(CLOCK_REALTIME, &ts);
ts.tv_sec += t;
int x = sem_timedwait(&electric, &ts);//wait till an electric stage is allocated or
waiting time exceeds t
if(x==-1){// if the waiting time exceeds t, then leave srujana
    printf("\033[1;32m%s %s left because of impatience\n\n\033[0m",inputs->name,
inputs->inst);
}
```

Next, we loop over all the electric stages to check which stage is free. The musician performs on that stage for some time between t1 and t2 and then checks if some singer had joined him during his performance. If yes, he further waits for 2 sec and then leaves the stage. He also unlocks es[l] which signals the singer performing with him (if any) to leave the stage. Further he signals all the semaphores for which he had waited.

```
sem_wait(&stage);//if an electric stage is allocated, we also need to increment the
stage semaphore
int l;
int k = random_func(t1,t2);
pthread_mutex_lock(&electric_stage);//to ensure that two musicians do not claim the
same stage simultaneously
//check which stage is free
for(int i=1; i<=e; i++){
    if(e_stage[i]==0){//check if any performer is performing of the stage
        e_stage[i]++;//increment the individual stage performer counter
        pthread_mutex_unlock(&electric_stage);//unlock the electric stage mutex
        l = i;
        // printf("%d %d\n\n", l, i);
        strcpy(electric_m_name[l], inputs->name);
        strcpy(electric_m_inst[l], inputs->inst);
```

```
        elec--;//decrement the number acoustic stages available
        break;
    }
}
printf("\033[1;35m%s performing %s at electric stage %d for %d sec\n\n\033[0m",
inputs->name, inputs->inst, l, k);
pthread_mutex_lock(&es[l]);
sleep(k);
if(e_stage[l]==2){//if a singer joins the musician during his performance wait for 2
sec
    sleep(2);
}
pthread_mutex_unlock(&es[l]);
printf("\033[1;34m%s's performance at electric stage %d finished\n\n\033[0m",inputs-
>name, l);
sem_post(&electric);//electric stage deallocated
sem_post(&stage);//if an electric stage is deallocated, we also need to decrement the
stage semaphore
elec++;//increment the number electric stages available
e_stage[l]=0;//decrement the individual stage performer counter
```

Next he collects tshirt from the coordinator

```
sem_wait(&tshirt);//wait for coordinator to be free to get the tshirt
printf("\033[1;36m%s collecting t-shirt\n\n\033[0m",inputs->name);
sleep(2);
sem_post(&tshirt);//leave the cooridnator so that he can attend other musicians
```

void both()

Musician arrives at Srujana as per the time given during input. He waits till any is allocated. If his waiting time exceeds t, he leaves Srujana.

```
struct timespec ts;
struct musician *inputs = (struct musician*)inp;
sleep(inputs->time);
printf("\033[1;37m%s %s arrived\n\n\033[0m",inputs->name,inputs->inst);
clock_gettime(CLOCK_REALTIME, &ts);
ts.tv_sec += t;
int x = sem_timedwait(&stage, &ts);//wait till a stage is allocated or waiting time
exceeds t
if(x==-1){// if the waiting time exceeds t, then leave srujana
    printf("\033[1;32m%s %s left because of impatience\n\n\033[0m",inputs->name,
inputs->inst);
}
```

Next we check which one of acoustic or electric stage is free. If both are free then we randomly choose any one of them, else we choose whichever is available.Then we loop over all the acoustic/electric (whichever is chosen) stages to check which stage is free. The musician performs on that stage for some time between t1 and t2 and then checks if some singer had joined him during his performance. If yes, he further waits for 2 sec and then leaves the stage. He also unlocks es[l]/as[l] which signals the singer performing with him (if any) to leave the stage. Further he signals all the semaphores for which he had waited.

```
char stg[50];
int flag=0;
if(astc > 0 && elec > 0){// if both electric and acoustic stages are available
    int h = rand()%2;
    if(h==0){// acoustic stage chosen
        strcpy(stg, "acoustic");
        sem_wait(&acoustic);//increment the acoustic semaphore
        flag=1;
        astc--;//decrement the number of acoustic stages available
    }
    else{//electric stage chosen
        strcpy(stg, "electric");
        sem_wait(&electric);//increment the electric semaphore
        flag=2;
        elec--;//decrement the number of electric stages available
    }
}
else if(astc==0){//if only electric stages are available
    strcpy(stg, "electric");
    sem_wait(&electric);//increment the electric semaphore
    flag=2;
    elec--;//decrement the number of electric stages available
}
else if(elec==0){// if only acoustic stages are available
    strcpy(stg, "acoustic");
    sem_wait(&acoustic);//increment the acoustic semaphore
    flag=1;
    astc--;//decrement the number of acoustic stages available
}
int l;
if(flag==1){
    pthread_mutex_lock(&acoustic_stage);//to ensure that two musicians do not claim
the same stage simultaneously
    //check which stage is free
    for(int i=1; i<=a; i++){
        if(a_stage[i]==0){//check if any performer is performing of the stage
            a_stage[i]++;//increment the individual stage performer counter
            pthread_mutex_unlock(&acoustic_stage);//unlock the acoustic stage mutex
            l = i;
            strcpy(acoustic_m_name[l], inputs->name);
            strcpy(acoustic_m_inst[l], inputs->inst);
            pthread_mutex_lock(&as[l]);
            break;
        }
    }
}
else if(flag==2){
    pthread_mutex_lock(&electric_stage);//to ensure that two musicians do not claim
the same stage simultaneously
    //check which stage is free
    for(int i=1; i<=e; i++){
        if(e_stage[i]==0){//check if any performer is performing of the stage
```

```
            e_stage[i]++;//increment the individual stage performer counter
            pthread_mutex_unlock(&electric_stage);//unlock the electric stage mutex
            l = i;
            strcpy(electric_m_name[l], inputs->name);
            strcpy(electric_m_inst[l], inputs->inst);
            pthread_mutex_lock(&es[l]);
            break;
        }
    }
}
int k = random_func(t1,t2);
printf("\033[1;35m%s performing %s at %s stage %d for %d sec\n\n\033[0m", inputs-
>name, inputs->inst, stg, l, k);
sleep(k);
if((a_stage[l]==2 && flag==1) || (e_stage[l]==2 && flag==2)){//if a singer joins the
musician during his performance wait for 2 sec
    sleep(2);
}
printf("\033[1;34m%s's performance at %s stage %d finished\n\n\033[0m",inputs->name,
stg, l);
if(flag==1){
    pthread_mutex_unlock(&as[l]);
    astc++;
    sem_post(&acoustic);//acoustic stage deallocated
    a_stage[l]=0;
}
if(flag==2){
    pthread_mutex_unlock(&es[l]);
    elec++;
    sem_post(&electric);//electric stage deallocated
    e_stage[l]=0;
}
sem_post(&stage);//if an electric/acoustic stage is deallocated, we also need to
decrement the stage semaphore
```

Then he collects tshirt from the coordinator.

```
sem_post(&stage);//if an electric/acoustic stage is deallocated, we also need to
decrement the stage semaphore
sem_wait(&tshirt);//wait for coordinator to be free to get the tshirt
printf("\033[1;36m%s collecting t-shirt\n\n\033[0m",inputs->name);
sleep(2);
sem_post(&tshirt);//leave the cooridnator so that he can attend other musicians
```

void only_singers()

Musician arrives at Srujana as per the time given during input. He waits till any stage is allocated. If his waiting time exceeds t, he leaves Srujana.

```
struct timespec ts;
struct musician *inputs = (struct musician*)inp;
sleep(inputs->time);
printf("\033[1;37m%s %s arrived\n\n\033[0m",inputs->name,inputs->inst);
```

```
clock_gettime(CLOCK_REALTIME, &ts);
ts.tv_sec += t;
int x = sem_timedwait(&singer, &ts);//wait if all the stages have singers or waiting
time exceeds t
if(x==-1){// if the waiting time exceeds t, then leave srujana
    printf("\033[1;32m%s %s left because of impatience\n\n\033[0m",inputs->name,
inputs->inst);
}
```

Singer first chooses either acoustic or electric (if both are available) or the one which is available then iterates over all the stages of that category to check which is free for a singer. If the stage is empty he does a solo performance, else he performes with a musician untill the musician signals him to leave. Then he leaves the stage and signals all the semaphores for which he has wait.

```
char stg[50];
int flag=0;
if(astc_singer > 0 && elec_singer > 0){//if acoustic stages and electric stages have
vacancy for a singer
    int h = rand()%2;
    if(h==0){//acoustic stage chosen
        strcpy(stg, "acoustic");
        flag=1;
        astc_singer--;//decrement the acoustic singer vacancy counter
    }
    else{//electric stage chosen
        strcpy(stg, "electric");
        flag=2;
        elec_singer--;//decrement the electric singer vacancy counter
    }
}
else if(astc_singer==0){//if only electric stages have vacancy for singers
    strcpy(stg, "electric");
    flag=2;
    elec_singer--;//decrement the electric singer vacancy counter
}
else if(elec_singer==0){//if only electric stages have vacancy for singers
    strcpy(stg, "acoustic");
    flag=1;
    astc_singer--;//decrement the acoustic singer vacancy counter
}
int l;
int var=0; char nm[50];
if(flag==1){
    pthread_mutex_lock(&acoustic_stage);//to ensure that two musicians do not claim
the same stage simultaneously
    //check which stage is free
    for(int i=1; i<=a; i++){
        if(a_stage[i]!=2){//if no singer is present on the stage
            a_stage[i]++;//increment the individual stage performer counter
            pthread_mutex_unlock(&acoustic_stage);//unlock the acoustic stage mutex
            l = i;
```

```
            if(a_stage[i]==2 && strcmp(acoustic_m_inst[l],"s")!=0){//if some musician
(not singer) is already performing on the stage
                var=2;
                strcpy(nm, acoustic_m_name[l]);

            }
            else{// if no musician is performing of the stage
                var=1;
                astc--;//decrement the number of acoustic stages available
                strcpy(acoustic_m_inst[l], inputs->inst);
                strcpy(acoustic_m_name[l], inputs->name);
                sem_wait(&acoustic);//occupy the acoustic stage
                sem_wait(&stage);//if an acoustic stage is allocated, we also need to
increment the stage semaphore
            }
            break;
        }
    }
}
else if(flag==2){
    pthread_mutex_lock(&electric_stage);//to ensure that two musicians do not claim
the same stage simultaneously
    //check which stage is free
    for(int i=1; i<=e; i++){
        if(e_stage[i]!=2){//if no singer is present on the stage
            e_stage[i]++;//increment the individual stage performer counter
            pthread_mutex_unlock(&electric_stage);//unlock the acoustic stage mutex
            l = i;
            if(e_stage[i]==2 && strcmp(electric_m_inst[l],"s")!=0){//if some musician
(not singer) is already performing on the stage
                var=2;
                strcpy(nm, electric_m_name[l]);
            }
            else{
                var=1;
                elec--;//decrement the number of electric stages available
                strcpy(electric_m_inst[l], inputs->inst);
                strcpy(electric_m_name[l], inputs->name);
                sem_wait(&electric);//occupy the electric stage
                sem_wait(&stage);//if an electric stage is allocated, we also need to
increment the stage semaphore
            }
            break;
        }
    }
}
if(var==1){// if singer is performing a solo
    int k = random_func(t1,t2);
    printf("\033[1;35m%s performing %s at %s stage %d for %d sec\n\n\033[0m", inputs-
>name, inputs->inst, stg, l, k);
    sleep(k);
    printf("\033[1;34m%s's performance at %s stage %d finished\n\n\033[0m",inputs-
```

```
>name, stg, l);
    if(flag==1){
        astc++;//increment the available acoustic stage counter
        sem_post(&acoustic);//deallocate the acoustic stage
        a_stage[l]=0;//decrement the individual stage performer counter
        astc_singer++;//increment the available acoustic stage singer counter
    }
    if(flag==2){
        elec++;//increment the available electric stage counter
        sem_post(&electric);//deallocate the electric stage
        e_stage[l]=0;//decrement the individual stage performer counter
        elec_singer++;//increment the available elctric stage singer counter
    }
    sem_post(&stage);//if an electric/acoustic stage is deallocated, we also need to
decrement the stage semaphore
}
else{// if singer is performing along with some musician
    printf("\033[01;33m%s joined %s's performance. Performance extended by 2
sec.\n\n\033[0m", inputs->name, nm);
    if(flag==1){
        pthread_mutex_lock(&as[l]);//wait till the musician signals to leave
        pthread_mutex_unlock(&as[l]);
    }
    if(flag==2){
        pthread_mutex_lock(&es[l]);//wait till the musician signals to leave
        pthread_mutex_unlock(&es[l]);
    }
    printf("\033[1;34m%s's performance at %s stage %d finished\n\n\033[0m",inputs-
>name, stg, l);
    if(flag==1){
        a_stage[l]=0;//decrement the individual stage performer counter
        astc_singer++;//increment the available acoustic stage singer counter
    }
    if(flag==2){
        e_stage[l]=0;//decrement the individual stage performer counter
        elec_singer++;//increment the available elctric stage singer counter
    }
}
sem_post(&singer);//decrement the singer semaphore
```

Then the singer collects tshirt from the coordinator.

```
sem_wait(&tshirt);//wait for coordinator to be free to get the tshirt
printf("\033[1;36m%s collecting t-shirt\n\n\033[0m",inputs->name);
sleep(2);
sem_post(&tshirt);//leave the cooridnator so that he can attend other musicians
```