Ober cab services

This code aims to simulate the cab services of a company called Ober

Assumptions Of the code:Cab drivers are of two types, Pool and Premier
Pool drivers can only accept cab fares of Pool types
Similarly for Premier

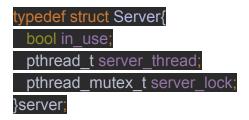
The maximum wait time, ride time and arrival time of the rider are random values between 1 to 10, 1 to 10 and 1 to 5 respectively

Approach

The code accepts the number of cabs, the number of riders and the number of payment servers that exist as input from the user. Drivers, Riders and Servers are the three main entities that are of concern in this code. Below you shall find the short description of each entity in the code with code snippets appropriately

Servers

The structural definition of servers is seen in the code snippet below



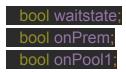
Here the boolean in_use indicates whether the respective server is in use or not An array of servers are created from this definition of the server.

Each of the servers have a mutex lock in order to ensure no occurrence of deadlocks and works perfectly in separate threads

Driver

The structural definition of the driver can be found below





bool onPool2;

pthread mutex t lock;

}driver;

Here the value of variable pno is the driver number, boolean variable type indicates whether the cab driver is Pool type driver or a Premier type driver.

Rider

Below is the structural definition of rider

```
typedef struct Rider{
  bool cabtype; ///0 if pool 1 if premier
  long int maxwait;
  int ridetime;
  int arrivaltime;
```

```
int pno;
 pthread_t rider_thread;
 pthread_mutex_t rider_lock;
 int s_no;
}rider;
```

Here cabtype indicates what type of cab the rider is in search of.

Code workings :-

Below following the code logic flow of the code

After appropriate initializations, each rider calls upon a rider function in their respective threads

```
for (int j = 1; j <= rider_no; ++j) { ///threads for each rider
    pthread_create(&r[j].rider_thread,NULL,rider_f,&r[j]);
}</pre>
```

After this the code for the function rider_f executes

In this function, we initially sleep for the arrival time sleep(current->arrivaltime);

The below loop searchs for an appropriate cab for the rider. If a cab is not found within the max wait time of the rider, the rider exits stating he/she has timed out

```
while(current->maxwait > timer && !booked){
i=1:
 for (; i <= cab_no && current->maxwait > timer; ++i) { ///look for a cab
 if (current->cabtype == d[i].type)
      if(current->cabtype){ ///premier type
         if(d[i].waitstate){
            bookcab(current,&d[i],&booked);
            if(booked)
             break;
      }else{///pool type
         if(d[i].onPool1 || d[i].waitstate){
            bookcab(current, &d[i], &booked);
           if (booked)
              break;
       pthread mutex unlock(&d[i].lock);
    time(&time2);
 timer=time2-time1; ///time elapsed in waiting mode
If the code exits without being booked, i.e having timed out, it is handled by the code snippet
below
if(!booked) {
printf("The passenger number %d has waited for too long and has hence Timed out
\n", current->pno);
return NULL;
Below is the function of bookcab which books the appropriate cab for the appropriate rider
void bookcab(rider *current, driver *pDriver,bool * booked) {
pthread mutex lock(&pDriver->lock); ///lock up the driver
```

```
if(current->cabtype){ ///its premier driver
    pDriver->onPrem=true;
    printf("Passenger %d is assigned driver %d for Premier
ride\n",current->pno,pDriver->pno);
    *booked=true;
 else if(current->cabtype==false){ ///its share driver
    if(pDriver->onPool1){
      pDriver->onPool2=true;
      pDriver->onPool1=false;
   } else{
      pDriver->onPool1=true;
   pDriver->onPool2=false;
  printf("Passenger %d is assigned driver %d for Pool
ride\n",current->pno,pDriver->pno);
    *booked=true;
 else{ ///false positives
  *booked=false;
pDriver->waitstate=false; ///Driver is booked and is not waiting
pthread mutex unlock(&pDriver->lock);
Once a cab has been booked
printf("Rider %d has started his ride with Driver %d\n",current->pno,i);
sleep(current->ridetime);
printf("Rider %d has finished his ride with Driver %d\n",current->pno,i);
initialize driver(i);
The ride time is simulated and the function initialize driver is called which basically resets the
details of the driver.
Below is the code implementation of initialize_driver
void initialize driver(int no) {
pthread mutex lock(&d[no].lock); ///lock up the driver
if(d[no].onPrem || d[no].onPool1){
    d[no].onPrem=false;
 d[no].waitstate=true;
```

```
d[no].onPool1=false;
   d[no].onPool2=false;
}
if(d[no].onPool2){
   d[no].onPool2=false;
   d[no].onPool1=true;
}
pthread_mutex_unlock(&d[no].lock); ///unlock the driver
```

After ride completion comes Payment using a server

Each server takes 2 seconds in total to process a payment request

The below is the code snippet from accept_payment function

```
void * accept_payment(void *arg) {
  rider * current=(rider *) arg;
  printf("Payment of rider %d is being processed \n",current->pno);
  sleep(2);
  printf("Payment of rider %d is completed \n",current->pno);
  s[current->s_no].in_use=false;
  return NULL;
}
```

The argument sent is the rider who is using the server at the moment This function is called as separate threads

The calling point of this function is present in line number 159 Line 159

pthread_create(&s[i].server_thread,NULL,accept_payment,current);

This thread call is bound on both sides by mutex lock and unlock of the concerned server to ensure thread safety

The code snippet showing the payment working is as follows



```
while(1){
    if(!s[i].in_use && (pthread_mutex_trylock(&s[i].server_lock)==0)){
        s[i].in_use=true;
        current->s_no=i;
        pthread_create(&s[i].server_thread,NULL,accept_payment,current);
        pthread_mutex_unlock(&s[i].server_lock);
        break;
    }
    i++;
    i%=server_no;
}
while (1){
    if(!s[i].in_use){ ///wait till the moment it frees up
        break;
    }
}
```

First we have a controlled infinite loop that keeps checking for a free server to use Once a server has been found, the payment process has begun and the code breaks of from this while loop and enters the next while loop which is an infinite loop until the server which is processing the payment of the rider becomes free. I.e the payment is over The second while loop is to ensure that the code does not finish before the payment for the driver is finished.(case occurs when there are lesser number of servers compared to drivers)

Result:-

The following is an example execution of the code

```
Enter number of cabs, riders and servers 2 10 3
Initializing driver 1 with cabtype 1
Initializing driver 2 with cabtype 1
Initializing rider 1 with cabtype 1, ridetime 2, maxwait time 3, arrival time 2
Initializing rider 2 with cabtype 0, ridetime 3, maxwait time 1, arrival time 2
Initializing rider 3 with cabtype 1, ridetime 3, maxwait time 4, arrival time 2
Initializing rider 4 with cabtype 0, ridetime 2, maxwait time 4, arrival time 1
Initializing rider 5 with cabtype 0, ridetime 2, maxwait time 4, arrival time 2
Initializing rider 6 with cabtype 1, ridetime 1, maxwait time 3, arrival time 1
Initializing rider 7 with cabtype 1, ridetime 1, maxwait time 3, arrival time 2
Initializing rider 8 with cabtype 0, ridetime 2, maxwait time 3, arrival time 2
Initializing rider 9 with cabtype 1, ridetime 2, maxwait time 2, arrival time 2
```

Initializing rider 10 with cabtype 0, ridetime 2, maxwait time 2, arrival time 2

Passenger 6 is assigned driver 1 for Premier ride

Rider 6 has started his ride with Driver 1

Passenger 3 is assigned driver 2 for Premier ride

Rider 3 has started his ride with Driver 2

Rider 6 has finished his ride with Driver 1

Passenger 7 is assigned driver 1 for Premier ride

Rider 7 has started his ride with Driver 1

Payment of rider 6 is being processed

The passenger number 2 has waited for too long and has hence Timed out

Rider 7 has finished his ride with Driver 1

Passenger 1 is assigned driver 1 for Premier ride

Rider 1 has started his ride with Driver 1

Passenger 9 is assigned driver 1 for Premier ride

Rider 9 has started his ride with Driver 1

Payment of rider 7 is being processed

The passenger number 10 has waited for too long and has hence Timed out

Payment of rider 6 is completed

The passenger number 8 has waited for too long and has hence Timed out

The passenger number 4 has waited for too long and has hence Timed out

Rider 3 has finished his ride with Driver 2

Payment of rider 3 is being processed

Rider 1 has finished his ride with Driver 1

Rider 9 has finished his ride with Driver 1

Payment of rider 1 is being processed

Payment of rider 7 is completed

Payment of rider 9 is being processed

The passenger number 5 has waited for too long and has hence Timed out

Payment of rider 3 is completed

Payment of rider 1 is completed

Payment of rider 9 is completed

All riders have been processed