```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>

int num; //number of passengers
struct station {

      pthread_mutex_t tpLock;
    pthread_cond_t trainArrived;
    pthread_cond_t passengerSettled;
    int boarded_passengers; //passengers boarded inside the train
      int passengers_inStation; //passengers waiting in the station
      int seats_vacant;       //vacant seats in the train

};
//FunctionDeclaration
int min(int x,int y);
void station_init(struct station *station);
void station_load_train(struct station *station, int count);
void station_wait_for_train(struct station *station);
void station_on_board(struct station *station);

//FunctionDefinition
void station_init(struct station *station)
{

      pthread_mutex_init(&station->tpLock , NULL); //initialize mutex
locks
      pthread_cond_init(&station->trainArrived ,NULL); //thread condition
variable
      pthread_cond_init(&station->passengerSettled,NULL);
      station->boarded_passengers=0;
    station->passengers_inStation = 0;
    station->seats_vacant = 0;

}

//train arrives/
void station_load_train(struct station *station, int count)
{
    //returns when there are no passengers or train is full

    pthread_mutex_lock(&station->tpLock);
    station->seats_vacant = count;
    while(station->seats_vacant > 0 && station->passengers_inStation > 0){

                            pthread_cond_broadcast(&station-
>trainArrived); //similar to used signal and used to inform several
threads which are waiting
            pthread_cond_wait(&station->passengerSettled , &station-
>tpLock);

    }
    station->seats_vacant = 0;
      pthread_mutex_unlock(&station->tpLock);
}


//passenger arrives

void station_wait_for_train(struct station *station)
{
    //return when there are enough available seats and train is in the
station
    pthread_mutex_lock(&station->tpLock);
      station->passengers_inStation++;

      while(station->boarded_passengers == station->seats_vacant){
        pthread_cond_wait(&station->trainArrived , &station->tpLock);
      }

      station->boarded_passengers++;
      station->passengers_inStation--;
      pthread_mutex_unlock(&station->tpLock);
}



//passenger boarded

void station_on_board(struct station *station)
{
    //to inform the train that it is on board
      pthread_mutex_lock(&station->tpLock);
      station->boarded_passengers--;
      station->seats_vacant--;

      if((station->seats_vacant == 0) || (station->boarded_passengers ==
0)){
        pthread_cond_signal(&station->passengerSettled);
      }

      pthread_mutex_unlock(&station->tpLock);

}

volatile int threads_completed = 0;
void* passenger_thread(void *arg)
{
      struct station *station=(struct station*) arg;
      station_wait_for_train(station);
        threads_completed++;
      return NULL;
}

struct TrainLoaded_Para {
      struct station *station;
      int free_seats;
};

volatile int return_LoadTrain = 0;

void* load_train_thread(void *args)
{
      struct TrainLoaded_Para *temp = (struct TrainLoaded_Para*)args;
      station_load_train(temp->station, temp->free_seats);
      return_LoadTrain = 1;
      return NULL;
}
//finds the minimum value among x and y
#ifndef MIN
#define MIN(_x,_y) ((_x) < (_y)) ? (_x) : (_y)
#endif
//main function starts from here
void main()
{
      struct station station;
      station_init(&station);

      srandom(getpid() ^ time(NULL)); //generates random numbers
      int i;
        printf("\n\n\n\t\t\tINDIAN RAILWAYS\n\n");
printf("\n\t\tNOTE*:NUmber of free seats in each train is initialized to 
60");
        printf("\n\n\tEnter the number of PASSSENGERS at the STATION : ");
        scanf("%d",&num);
      if(num<0)
      {
      printf(" \tYou have entered number of passengers as %d which is not 
possible.\n",num);
      printf(" \tPlease enter a valid number!!\n");
      scanf("%d",&num);
      }
      if(num==0)
      {
      printf(" \t NO PASSENGERS in the STATION!!\n\n");
      exit(0);
     }
      const int total_Passngrs=num;
      int remaining_Passngrs = total_Passngrs;
      for (i = 0; i < total_Passngrs; i++) {
                  pthread_t tid;
                  int ret = pthread_create(&tid, NULL, passenger_thread, 
&station);
                  }

       int total_Passngrs_boarded = 0;
      const int tot_FreeSeats_PerTrain =100;
      int pass = 0;
        int j=1,p=1;
```

```
while (remaining_Passngrs > 0) {

                int free_seats = random() % tot_FreeSeats_PerTrain;

                printf("  \tTRAIN[ %d ] has entered the STATION : Free
SeatsAvailable - %d\n\n",j,free_seats);
               j++;
                return_LoadTrain = 0;
                struct TrainLoaded_Para args = { &station, free_seats };
                pthread_t lt_tid;
                int ret = pthread_create(&lt_tid, NULL,
load_train_thread, &args);
                if (ret != 0) {
                           perror("pthread_create");
                           exit(1);
                }

                int threads_to_reap = MIN(remaining_Passngrs,
free_seats);
                int threads_reaped = 0;

                while (threads_reaped < threads_to_reap) {

if(return_LoadTrain){
exit(1);
}
                           if (threads_completed > 0) {
                                if ((pass % 2) == 0)
                                          usleep(random() % 2);
                                threads_reaped++;
                                station_on_board(&station);
                              threads_completed++;

                           }
                }

                remaining_Passngrs -= threads_reaped;
                total_Passngrs_boarded += threads_reaped;
                printf("  \tTRAIN[ %d ] DEPARTED the STATION    :   New
Passengers    - %d  :\n\n",
                           p,threads_to_reap);

                pass++;
               p++;
     }

     if (total_Passngrs_boarded == total_Passngrs) {
                printf("\t\t\t ALL PASSENGERS BOARDED!\n");
     }
     exit(0);
}
```