#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

void \*car\_thread (void \*p)

{

struct cars\_on\_bridge ;

pthread\_mutex\_lock(&m);

while(cars\_on\_bridge > 5 || (dir\_on\_bridge!=-1 && dir\_on\_bridge != car->dir))

pthread\_cond\_wait(cq[car->dir], &m)

cars\_on\_bridge++;

dir\_on\_bridge = car->dir;

pthread\_mutex\_unlock(&m);

usleep(5000000);

pthread\_mutex\_lock(&m);

cars\_on\_bridge--;

if (cars\_on\_bridge > 0)

{

pthread\_cond\_signal(cq[car->dir]);

}

else

{

dir\_on\_bridge = -1;

pthread\_cond\_broadcast(cq[1-car->dir]);

}

pthread\_mutex\_unlock(&m);

free(car);

}

int main ()

{

pthread\_attr\_t attr;

pthread\_attr\_init(&attr);

pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_DETACHED);

while (1)

{

car = malloc(sizeof(struct CarInfo));

car->id = ++car\_id;

car->dir = rand() & 1;

pthread\_create(&thr\_id, &attr,

car\_thread, (void \*) car)

unsleep(200000);

}

return 0;

}

struct bridge

{

int north\_waiting;

int north\_crossing;

int north\_consecutive;

int south\_waiting;

int south\_crossing;

int south\_consecutive;

struct lock \*lock;

struct condition \*northbound\_done;

struct condition \*southbound\_done;

};

struct lock \*lock;

void bridge\_init(struct bridge \*b)

{

b->north\_waiting = 0;

b->north\_crossing = 0;

b->north\_consecutive = 0;

b->south\_waiting = 0;

b->south\_crossing = 0;

b->south\_consecutive = 0;

lock\_init(&b->lock);

cond\_init(&b->northbound\_done);

cond\_init(&b->southbound\_done);

}

int bridge\_arrive\_north(struct bridge \*b)

{

lock\_acquire(&b->lock);

b->north\_waiting++;

while ((b->south\_crossing > 0) ||(b->south\_waiting > 0) && (b->north\_consecutive >= 5))

{

cond\_wait(&b->southbound\_done);

}

b->north\_waiting--;

b->north\_crossing++;

b->north\_consecutive++;

b->south\_consecutive = 0;

lock\_release(&b->lock);

}

int bridge\_leave\_north(struct bridge \*b)

{

lock\_acquire(&b->lock);

b->north\_crossing--;

if (b->north\_crossing == 0)

{

cond\_broadcast(b->northbound\_done);

}

lock\_release(&b->lock);

}

int bridge\_arrive\_south(struct bridge \*b)

{

lock\_acquire(&b->lock);

b->south\_waiting++;

while((b->north\_crossing > 0) || ((b->north\_waiting > 0) && (b->south\_consecutive >= 5)))

{

cond\_wait(&b->northbound\_done);

}

b->south\_waiting--;

b->south\_crossing++;

b->south\_consecutive++;

b->north\_consecutive = 0;

lock\_release(&b->lock);

}

int bridge\_leave\_south(struct bridge \*b)

{

lock\_acquire(&b->lock);

b->south\_crossing--;

if (b->south\_crossing == 0)

{

cond\_broadcast(b->southbound\_done);

}

lock\_release(&b->lock);

}