// Created by Ayoola Etiko on 2020-05-25.

// Copyright © 2020 Ayoola Etiko. All rights reserved.

//

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

#include <time.h>

#include <pthread.h>

#include <semaphore.h>

typedef struct car{

char myName[25];

int myDir;

int myArrive;

int myDur;

int myIndex;

} car;

static sem\_t bridge[10];

static sem\_t indexLock;

static sem\_t dirLock;

static sem\_t tLock;

int ind;

int direction;

int t;

int numCars;

void \* routeTaken(void \* cur);

car \* carInit(char \* nameIn, int dirIn, int arriveIn, int durationIn);

int main(){

ind = 0;

direction = -1;

numCars = 0;

t = 0;

int i;

int j;

for(i=0;i<10;i++){

sem\_init(bridge+i, 0, 0);

}

char user[20];

char direction[1];

//help to get rid of first line in code instead of using stdin

char garbage[10];

int arrival = 0;

int duration = 0;

int ep = 0;

car \* list;

if((list = malloc(10 \* sizeof(car))) == NULL){

return EXIT\_FAILURE;

}

int z;

for(z=0;z<4;z++){

scanf("%s\t", &garbage);

}

int safe = 0;

while(safe < 4 && (scanf("%s\t%s\t%d\t%d",&user, &direction, &arrival, &duration)==4)){

car \*newInstance;

int conv = -1;

if(strcmp(direction, "N") == 0){

conv = 1;

}else{

conv = 0;

}

newInstance = carInit(user, conv, arrival, duration);

list[ep] = \*newInstance;

ep++;

safe++;

}

pthread\_t pileUp[ep];

for(j=0;j<ep;j++){

car \* carPtr;

carPtr = list+j;

pthread\_create(&pileUp[j], NULL, &routeTaken, (void\*)carPtr);

}

int g;

for(g=0; g<ep; g++){

pthread\_join(pileUp[g],NULL);

}

return EXIT\_SUCCESS;

}

void \* routeTaken(void \* ptrToCurrentCar){

car \* carIn = (car\*)ptrToCurrentCar;

int onBridge = 0;

int locT;

while(carIn->myDur != 0){

locT = t;

if(onBridge == 0){

if(carIn->myArrive < t){

if(direction == -1){

sem\_post(&bridge[ind]);

sem\_post(&indexLock);

carIn->myIndex = ind;

sem\_post(&dirLock);

direction = carIn->myDir;

sem\_post(&dirLock);

onBridge = 1;

numCars++;

if(carIn->myDir == 1){

printf("Direction: North\n");

}else{

printf("Direction: South\n");

}

}else{

if(carIn->myDir == direction){

sem\_post(&bridge[ind+1]);

carIn->myIndex = ind+1;

onBridge = 1;

ind++;

numCars++;

}else{

sem\_post(&bridge[ind]);

sem\_wait(&bridge[ind]);

}

}

}

}else{

carIn->myDur--;

}

locT = t;

locT++;

sem\_post(&tLock);

t = locT;

sem\_wait(&tLock);

}

if(carIn->myIndex != 0){

sem\_post(&indexLock);

}

sem\_wait(&dirLock);

numCars--;

printf("Name: %s\n", carIn->myName);

sem\_post(&indexLock);

if(numCars == 0){

direction = -1;

ind = 0;

}

sem\_wait(&bridge[carIn->myIndex]);

sem\_wait(&dirLock);

return 0;

}

car \* carInit(char \* user, int direction, int arrival, int duration){

car \* tempCar;

if((tempCar = malloc(sizeof(car))) == NULL){

return NULL;

}

strcpy(tempCar->myName, user);

tempCar->myDir = direction;

tempCar->myArrive = arrival;

tempCar->myDur = duration;

tempCar->myIndex = -1;

return tempCar;

}