```c
/*
 * INTRODUCTION:
 * The purpose of this lab is to design and test a program that use multi-threading with synchronization mechanis,s.
 *
 * DESIGN:
 * -two structs, One struct for simulation parameters and the other struct is for car data
 * -Used a Semaphore for the construction zone or road.
 * -sem wait and sem post 3 times to ensure a maxmimum of 3 cars available on the street.
 * -safe for a car to drive if car direction is equal to flagger direction
 * -if it is not safe, it has to wait, which means it has to poll
 * -flagger keeps flipping direction
 *
 * BUILD INSTRUCTIONS:
 * build: gcc -g flagger ahead.c -o flagger -pthread
 * Run: ./flagger parameters | less
 *
 * ANALYSIS:
 * the calculated time looks fair, as I make sure each moves out the street.
 * time is started before the cars enters the road and ends after it crosses
 * My implementation still has problems, sometimes when the flagger flips direction there are no cars moving
 * if this problem was fixed, i think I meet all the requirements of the lab
 *
 * CONCLUSION:
 * It was a very challenging lab, it took a lot of time debugging.
 * usage of semaphores and clock, helped a lot.
 *
 *
 *
 *
 *
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <sys/types.h>
#include <semaphore.h>
#include <unistd.h>
#include <stdbool.h>


//Struct Definitions
typedef struct{
    int n cars west dir;//No of cars on the west end of Construction zone
    int n cars east dir;//No of cars on the east end of Construction zone
    int car drive time;//amount of time taken to drive through the construction zone
    int flagger switch time;//time after which the flags must be switched
    int max cars crossing;//maximum number of cars crossing in one direction
}simulation_parameters;

typedef struct{
    int direction;
    int car number;
    int n car crossings;
    int car wait time;
    int time to cross;
    int car total wait time;
}car_data;

```

```
62  //Function Definitions
63  void read simulation parameters(FILE *,simulation_parameters *, char *argv[]);
64  void read car data(FILE *,int, car_data *, char *argv[]);
65  void create threads(simulation_parameters *,car_data *);
66
67  //Global variables
68  #define WEST TO EAST 0
69  #define EAST TO WEST 1
70  #define initial direction WEST TO EAST
71  sem t road;
72  int flagger direction = WEST TO EAST;
73  int flag = 0;
74  struct timespec start, end, temp;
75
76  int main(int argc, char* argv[]){
77
78      sem init(&road,0,3);
79
80      simulation_parameters *simulation data = malloc(sizeof(simulation_parameters));
81
82      FILE* simulator= fopen(argv[1],"r");
83      read simulation parameters(simulator,simulation data,argv);
84
85
86      car_data *car metrics =                                                    ↵
        malloc(simulation data->n cars west dir*simulation data->n cars east dir*sizeof(ca ↵
        r_data));
87
88
89                                                                                 ↵
        read car data(simulator,(simulation data->n cars west dir+simulation data->n cars  ↵
        east dir), car metrics, argv);
90
91      for(int i=0;i<simulation data->n cars west dir;i++){
92          car metrics[i].direction = WEST TO EAST;
93      }
94      for(int                                                                    ↵
        i=simulation data->n cars west dir;i<simulation data->n cars west dir+simulation d ↵
        ata->n cars east dir;i++){
95          car metrics[i].direction = EAST TO WEST;
96      }
97
98      car metrics->time to cross = simulation data->car drive time;
99      printf("\nCONSTRUCTION ZONE PARAMETERS\n");
100     printf("INTIAL CARS WEST END %d ",simulation data->n cars west dir);
101     printf("EAST END %d\n",simulation data->n cars east dir);
102     printf("CONSTRUCTION ZONE CROSS TIME %d ",simulation data->car drive time);
103     printf("CAR CAPACITY %d\n",simulation data->max cars crossing);
104     printf("FLAGGER FLIP TIME BEFORE FLIPPING DIRECTION:%d                     ↵
        \n",simulation data->flagger switch time);
105
106
107     for(int i = 0; i < (simulation data->n cars west dir); i++){
108     printf("Car %d - Crossing Count:%d ",i,car metrics[i].n car crossings);
109     printf("Sleep Time:%d ",car metrics[i].car wait time);
110     printf("Initial Direction: WEST->EAST\n");
111     }
112     for(int                                                                    ↵
        i=simulation data->n cars west dir;i<simulation data->n cars west dir+simulation d ↵
        ata->n cars east dir;i++){
113     printf("Car %d - Crossing Count:%d ",i,car metrics[i].n car crossings);
114     printf("Sleep Time:%d ",car metrics[i].car wait time);
115     printf("Initial Direction: EAST->WEST\n");
116     }
117
```

```
118         for(int i=0;
            i<(simulation data->n cars west dir+simulation data->n cars east dir); i++){
119                 car metrics[i].car number = i;
120         }
121         create threads(simulation data,car metrics);
122
123         printf("Simulation Finished\n");
124
125         for(int i=0;
            i<(simulation data->n cars west dir+simulation data->n cars east dir); i++){
126                 printf("Car %d waited a total of
                    %dns\n",i,car metrics[i].car total wait time);
127         }
128
129         fclose(simulator);
130         free(simulation data);
131         free(car metrics);
132     }
133
134     /*
135      * This Function reads the simulation parameters
136      */
137     void read simulation parameters(FILE *simulator,simulation parameters
        *simulation data,char *argv[]){
138
139         int n cars west dir;//No of cars on the west end of Construction zone
140         int n cars east dir;//No of cars on the east end of Construction zone
141         int car drive time;//amount of time taken to drive through the construction zone
142         int flagger switch time;//time after which the flags must be switched
143         int max cars crossing;//maximum number of cars crossing in one direction
144
145
146         fscanf(simulator,"%d",&n cars west dir);
147         fscanf(simulator,"%d",&n cars east dir);
148         fscanf(simulator,"%d",&car drive time);
149         fscanf(simulator,"%d",&flagger switch time);
150         fscanf(simulator,"%d",&max cars crossing);
151
152         simulation data->n cars west dir = n cars west dir;
153         simulation data->n cars east dir = n cars east dir;
154         simulation data->car drive time = car drive time;
155         simulation data->flagger switch time = flagger switch time;
156         simulation data->max cars crossing = max cars crossing;
157
158     }
159     void read car data(FILE *simulator,int no of cars, car_data *n car data, char *argv[]){
160
161         int n car crossings;
162         int car wait time;
163
164         for(int i=0; i<no of cars;i++){
165             fscanf(simulator,"%d",&n car crossings);
166             fscanf(simulator,"%d",&car wait time);
167             n car data[i].n car crossings = n car crossings;
168             n car data[i].car wait time = car wait time;
169         }
170     }
171
172     void cross street(car_data *car metrics){
173         // poll until direction changes
174         while(flagger direction != car metrics->direction);
175         //printf("Car %d left road!\n", car metrics->car number);
176         // cross road
177         clock gettime(CLOCK REALTIME,&start);
178         sem_wait(&road);
```

```
179         if(car metrics->direction == WEST TO EAST){
180             printf("Car %d entering construction zone traveling: WEST->EAST crossing
                remaining %d\n",car metrics->car number,car metrics->n car crossings);
181         }
182         else {
183             printf("Car %d entering construction zone traveling: EAST->WEST crossing
                remaining %d\n",car metrics->car number,car metrics->n car crossings);
184         }
185         usleep(car metrics->time to cross);
186         clock gettime(CLOCK REALTIME,&end);
187         sem post(&road);
188         //time calculation
189                 if((end.tv nsec-start.tv nsec) < 0 ){
190                     temp.tv sec = end.tv sec-start.tv sec-1;
191                     //  1/10^9 per second
192                     temp.tv nsec = (1000000000+ end.tv nsec-start.tv nsec);
193                 }
194                 else {
195                     temp.tv sec = end.tv sec-start.tv sec;
196                     temp.tv nsec = end.tv nsec-start.tv nsec;
197                 }
198                 car metrics->car total wait time = temp.tv nsec;
199         //printf("Car %d left road!\n", car metrics->car number);
200     }
201     void *car thread(void *args){
202         car data *car metrics = (car data*) args;
203         while(car metrics->n car crossings > 0) {
204             cross street(car metrics);
205             car metrics->n car crossings--;
206             if(car metrics->n car crossings > 0){
207                 usleep(car metrics->car wait time);
208                 if(car metrics->direction == EAST TO WEST){
209                     //printf("CAR EAST TO WEST\n");
210                 //  printf("Car %d entering construction zone traveling: WEST TO EAST
                    crossing remaining
                    %d\n",car metrics->car number,car metrics->n car crossings);
211                     car metrics->direction = WEST TO EAST;
212                 } else {
213                     //printf("CAR WEST TO EAST\n");
214                 //  printf("Car %d entering construction zone traveling: EAST TO WEST
                    crossing remaining
                    %d\n",car metrics->car number,car metrics->n car crossings);
215                     car metrics->direction = EAST TO WEST;
216                 }
217             }
218         }
219         pthread exit(0);
220     }
221
222     void *flagger thread(void* args){
223         simulation parameters *simulation data = (simulation parameters*) args;
224         //printf("SUCESSSS2\n");
225         //printf("I'M HERE456");
226         flagger direction = initial direction;
227         while(true){
228             //printf("I'M HERE");
229
230             usleep(simulation data->flagger switch time);
231
232             sem wait(&road);
233             sem wait(&road);
234             sem wait(&road);
235
236             if(flagger direction == WEST TO EAST){
237                 printf("Flagger Indicating Safe to Drive EAST_TO_WEST\n");
```

```
238                 flagger direction = EAST TO WEST;
239             } else {
240                 printf("Flagger Indicating Safe to Drive WEST TO EAST\n");
241                 flagger direction = WEST TO EAST;
242             }
243 
244             sem post(&road);
245             sem post(&road);
246             sem post(&road);
247         }
248     }
249 
250 
251 
252     void create threads(simulation parameters *simulation data, car data *car metrics){
253         pthread t car threads[simulation data->n cars east dir *
        simulation data->n cars west dir];
254         pthread t thread for flagger;
255 
256 
257 
        if(pthread create(&(thread for flagger),NULL,&flagger thread,((void*)simulation da
        ta))){
258         printf("error");
259         }
260 
261         for(int i=0;
        i<(simulation data->n cars west dir+simulation data->n cars east dir); i++){
262 
263 
            if(pthread create(&(car threads[i]),NULL,&car thread,((void*)(&car metrics[i])
            ))){
264                 printf("Error");
265             }
266         }
267 
268         for(int i=0;
        i<(simulation data->n cars west dir+simulation data->n cars east dir);i++){
269             pthread join(car threads[i],NULL);
270         }
271 
272     }
273 
274 
```