Thi Nguyen
CS475 A2 Part B

One Vehicle Design Document

Policy 1:

To implement the default policy, besides the build in variables and data structures, I declared the following

```
int waiters[2];  /* And array of two int that keeps track of the number of cars waiting to go in each East and West direction */
int eastQueue[30]; /* FIFO queue that keeps the waiting cars want to go east */
int headEQ = 0;  /* the index of head of east queue */
int tailEQ = 0;          /* the index of tail of east queue */
int westQueue[30]; /* FIFO queue that keeps the waiting cars want to go west */
int headWQ = 0;  /* the index of head of west queue */
int tailWQ = 0;          /* the index of tail of west queue */
int numHaveCrossed = 0;  /* number Of Vehicles Have Crossed In Cur Direc */

pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t waitingToGoEast = PTHREAD_COND_INITIALIZER;
pthread_cond_t waitingToGoWest = PTHREAD_COND_INITIALIZER;
```

ArriveBridge:

To implement this policy, in the ArriveBridge function, I initially acquire lock because I will have to modify the shared variables such as the bridge structure and others declared above. Once the car arrive, it will wait in the queue if the following condition happens,

```
while ((br.num_car == 5) || (br.num_car > 0 && br.curr_dir != direc))
```

That is, the current number of cars on the bridge reaches limit of 5 or when there is car crossing in the opposite direction. If so, the car will be placed in the queue. As follow

```
waiters[direc]++;
if (direc == 0) {
        eastQueue[tailEQ] = vid;
        tailEQ++;
```

After it is added to its queue, it will wait for signal from ExitBridge,

```
        pthread_cond_wait(&waitingToGoEast,&lock);
```

Once the thread receive signal to wake up to cross the bridge, it still has to wait till its turn because the signal was sent to all threads. So if there is any vehicle before it its queue, it has to wait:

```
        while (eastQueue[headEQ] != vid)
                pthread_cond_wait(&waitingToGoEast,&lock);
        headEQ++;
```

Similarly for the car in the west direction

```
else {
```

```
                westQueue[tailWQ] = vid;
                tailWQ++;
                pthread_cond_wait(&waitingToGoWest,&lock);
                /* Once the thread receive signal to wake up to cross the bridge,
                        It still has to wait till its turn. So if there is any vehicle
                        before it its queue, it has to wait
                 */
                while (westQueue[headWQ] != vid )
                        pthread_cond_wait(&waitingToGoWest,&lock);
                headWQ++;
        }
```

Once the car is woke up we decrease the number of waiter in that queue.

```
        waiters[direc]--;
        }
```

Once the car hits to this point of the code, it is about to get on the bridge, and I need to modify the following variables to indicate the event:

```
        /* Car get on the bridge */
        br.num_car++;
        br.curr_dir = direc;
```

And then release the lock:

```
        pthread_mutex_unlock(&lock);
```

ExitBridge:

I also initially acquire lock, and to indicate the fact the car gets car gets off the bridge, I modify the following:

```
        br.num_car--;
        br.dept_idx++;
```

Then I will check if anybody wants to go on the same direction of the bridge curr_dir, wake them up

```
        if (waiters[br.curr_dir] > 0) {
                if (br.curr_dir == 0) { pthread_cond_signal(&waitingToGoEast); }
                else { pthread_cond_signal(&waitingToGoWest); }
        }
```

Otherwise, if no other car is waiting to pass in the curr_dir and there is no car on the bridge wake up the cars waiting in the other side using broadcast :

```
        else if (br.num_car == 0) {
                if (br.curr_dir == 0) { pthread_cond_broadcast(&waitingToGoWest); }
                else { pthread_cond_broadcast(&waitingToGoEast); }
        }
```

Then I release the lock: pthread_mutex_unlock(&lock);

Policy 2:

Similarly to policy 1, but I added a variable numHaveCrossed to indicate the number Of Vehicles Have Crossed In Cur Direction.
I figure that the condition for a car to wait when it arrives the bridge is different:

```
while ((br.num_car == 6) || (br.curr_dir != direc && numHaveCrossed < 6) ||
        (br.curr_dir == direc && numHaveCrossed == 6 && waiters[1-direc]))
```

That is, the current number of cars on the bridge is 6 or when the current direction is not the direction of the car and the number of car have crossed is less than 6 or when the current direction is the same as the car direction

And if the current direction in the bridge is not the same as the bridge direction, (br.curr_dir != direc), I let the car wait till all 6 cars have crossed the bridge and then I allow the car to cross and switch the bridge direction. Once the bridge switches direction, set the numHaveCrossed to 0