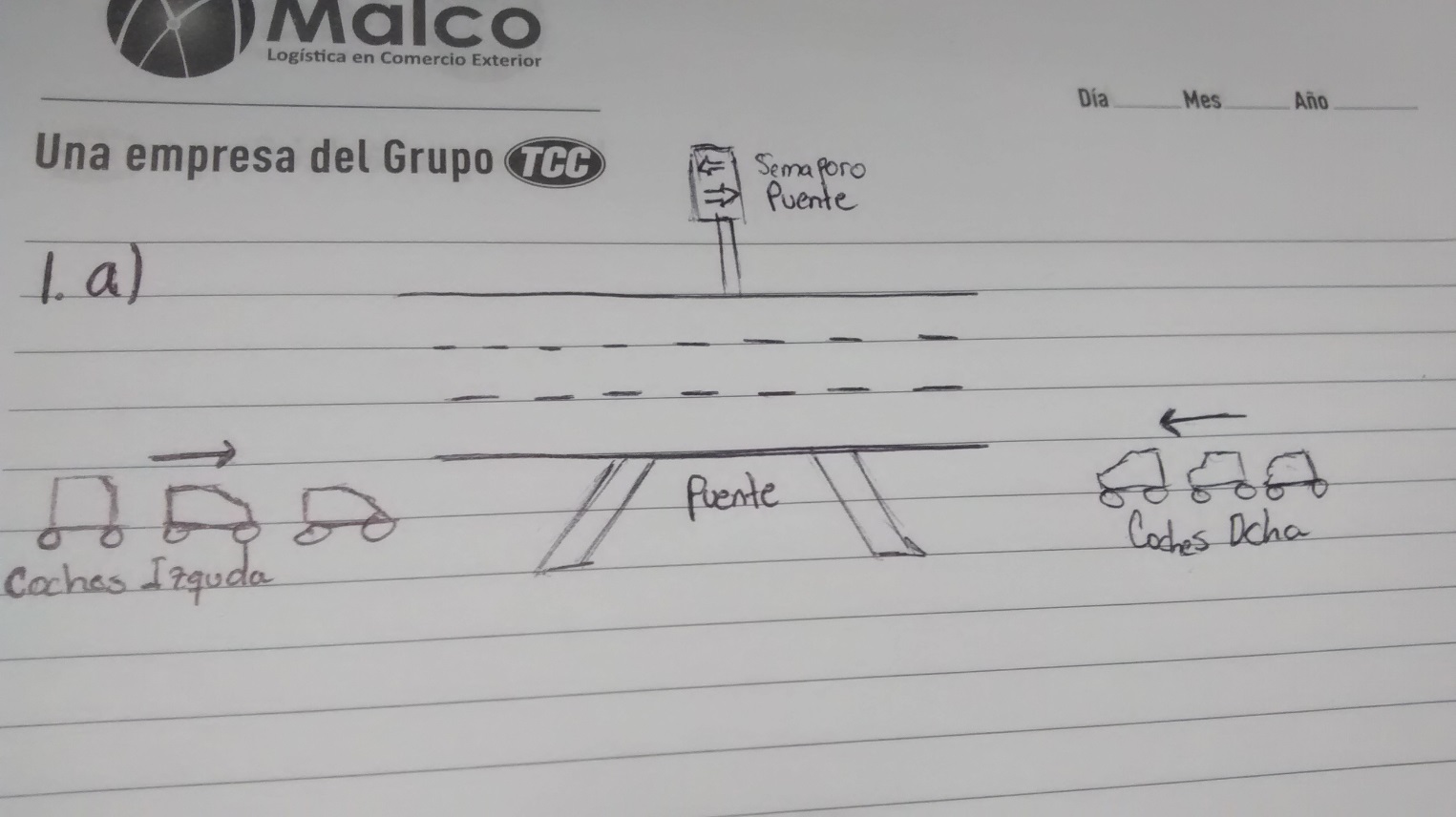
Juan Camilo Amézquita – 1220300

URL del Repositorio: https://github.com/camiloamezquita95/icesi\_examen\_1

**Parcial 1 Sistemas Operativos**



* 1. **Semáforo Espacios Disponibles:** Cantidad de espacios disponibles en el puente, vela por que la cantidad de coches en el puente simultáneamente no sean más de 10.

El semáforo del puente que da paso a los coches a través del método vaciarPuente realiza V() para aumentar la cantidad de espacios disponibles en el puente una vez hayan cruzado.

**Semáforo Coches Disponibles Derecha:** Cantidad de coches al lado derecho del puente que están listos para cruzar el puente.

Los coches a través del método colocarVehiculosDerEnPuente de la clase Puente realizan V() ya que aumentan el semáforo en 1 indicando que existe un nuevo coche listo para cruzar el puente de derecha a izquierda.

**Semáforo Coches Disponibles Izquierda:** Cantidad de coches al lado izquierdo del puente que están listos para cruzar el puente.

Los coches a través del método colocarVehiculosIzqEnPuente de la clase Puente realizan V() ya que aumentan el semáforo en 1 indicando que existe un nuevo coche listo para cruzar el puente de izquierda a derecha.

**Semáforo mutex:** Semáforo que garantiza que las variables sean escritas sincronizadamente.

Tanto la clase CochesIzq como la clase CochesDer realizan las operaciones P () y V ().

P () antes de escribir o leer las variables compartidas y V () cuando ya han sido leídas o escritas.

**Semáforo mutex coches derecha:** Semáforo que garantiza que las variables usadas por los coches del lado derecho sean escritas sincronizadamente.

La clase CochesDer realiza las operaciones P () y V ().

P () antes de escribir o leer las variables compartidas y V () cuando ya han sido leídas o escritas.

**Semáforo mutex coches izquierda:** Semáforo que garantiza que las variables usadas por los coches del lado izquierdo sean escritas sincronizadamente.

La clase CochesIzq realiza las operaciones P () y V ().

P () antes de escribir o leer las variables compartidas y V () cuando ya han sido leídas o escritas.

**Semáforo Puente:** Clase que se encarga de dar paso a los vehículos en una única dirección, realiza los cambios de vía cada cierto tiempo de manera aleatoria para garantizar que no exista inanición.

Al llamar el método darPaso de la clase puente realiza las operaciones P () y V () de los semáforos dependiendo de la orientación que tenga en ese momento la via, es decir si es de derecha a izquierda o de izquierda a derecha.

**Maximización de Recursos:**

La solución planteada maximiza los recursos de buena manera ya que la clase SemaforoPuente realiza un cambio de orientación del puente (izquierda a derecha o derecha a izquierda) cada cierto tiempo aleatorio, además de esto en el caso en el que se le de paso a un lado donde no existan vehículos la orientación del puente cambia automáticamente para darle paso a los vehículos del otro lado. Por esto el puente (Recurso) siempre tendrá vehículos y será usado al máximo para dejar pasar la mayor cantidad de vehículos posible sin que exista inanición en ninguno de ambos lados.

**Evitar la Inanición:**

Debido a que como se dijo en el punto anterior el paso de vehículos se alterna cada cierto tiempo aleatorio, nunca se quedaran vehículos en alguno de los dos lados esperando para poder cruzar, no obstante en el caso de que no hayan carros en cierto momento y se le de vía al otro lado y lleguen vehículos rápidamente al lado donde inicialmente no habían vehículos tendrán que esperar hasta que sea su turno para cruzar. Este mecanismo evita la inanición por completo, ya que funciona como un semáforo en la vida real pero con la pequeña variación de que este semáforo es inteligente, porque al determinar la ausencia de vehículos en un lado procede a cambiar el sentido para darle paso a los vehículos del otro lado.

* 1. Revisar el proyecto llamado Parcial1SistemasOperativosF que está en el repositorio.

<https://github.com/camiloamezquita95/icesi_examen_1>

* 1. Caso:

Los generadores de coches CochesDer y CochesIzq generan vehículos en su lado respectivo cada cierto tiempo aleatorio. Las dos clases mencionadas anteriormente se encargan de pasar a cada lado del puente los vehículos generados usando los métodos colocarVehiculosDerEnPuente() y colocarVehiculosIzqEnPuente() respectivamente, estos métodos usan el semáforo coches disponibles derecha y semáforo coches disponibles izquierda usando release() para aumentar los recursos en un 1, además de esto también llaman a sus mutex respectivos que bloquean las variables compartidas en este caso cantidadCochesDer y cantidadCochesIzq, para este caso digamos que se generaron en 1 minuto 23 carros a la izquierda y 15 a la derecha.

Al tener vehículos a ambos lados la clase SemaforoPuente da vía en el puente cada cierto tiempo aleatorio, la clase Puente se encarga de darle vía a los vehículos del lado del cual es el turno para cruzar, para este caso digamos que el sentido de la via es de Izquierda a derecha.

Al darle paso a los vehículos del lado izquierdo el puente realiza acquire() del semáforo coches disponibles izquierda y acquire() del semáforo encargado de garantizar que no hayan más de 10 vehículos en el puente llamado semáforo espacios disponibles y por último, procede a vaciar el puente de vehículos y pasarlos al otro lado, además de esto se realizan los reléase() de los recursos, en este caso el semáforo espacios disponibles, indicando que el puente ya puede recibir nuevos vehículos.

Por último se realiza el cambio del sentido de la vía del puente para que crucen los vehículos de derecha a izquierda.

Al realizar este algoritmo pasaria lo siguiente: primero pasaran 10 vehículos del lado izquierdo (ya que es la cantidad máxima de vehículos que pueden estar sobre el puente), luego cruzaran 10 vehículos de derecha a izquierda, de nuevo cruzaran 10 vehículos de la izquierda y únicamente los 5 vehículos restantes de la derecha, por ultimo cruzaran los 3 vehículos restantes que estaban en el lado izquierdo.

Simultáneamente todos los coches que están subiendo al puente usan las variables compartidas de manera sincronizada gracias a los mutex correspondientes.

1. Repositorio Subido con éxito, el repositorio contiene el proyecto y el documento requerido.

<https://github.com/camiloamezquita95/icesi_examen_1>

1. Fork Realizado al repositorio de Daniel Lora.

URL: https://github.com/camiloamezquita95/icesi\_examen\_1-1