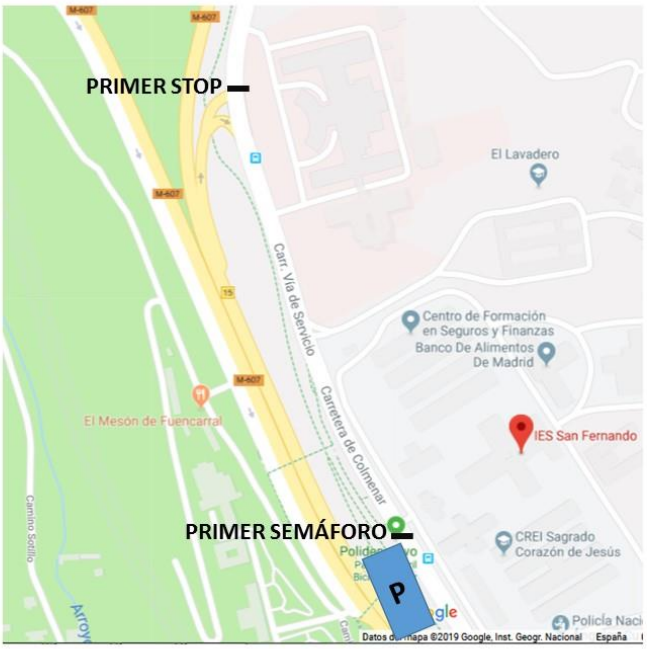
Programación II. Parcial 12 de Marzo 2019

Apellidos: _____Nombre:_____

Ejercicio 5. (3 puntos) Formas parte de un equipo de programadores que está desarrollando una aplicación de monitorización y gestión del tráfico de una ciudad. Actualmente estáis implementando un simulador que posee un mapa con los semáforos y stops que hay en dicho mapa. En concreto, os estáis centrando en la zona que va desde la salida sur de la UAM hasta el primer aparcamiento del IES San Fernando.

En esa zona, vamos a suponer que un vehículo que saliera de la UAM hacia el instituto se encontraría 1 STOP y 1 semáforo antes de llegar al aparcamiento (ver figura). A la hora de salida del instituto, a menudo se forman atascos en la vía de servicio (solo en la zona del STOP y del semáforo, no en la salida de la UAM).



Escribe el **pseudocódigo** de una función que simule el avance de un vehículo desde la UAM hasta el aparcamiento del instituto. El vehículo saldrá de la UAM y recorrerá la vía de servicio sin problemas hasta llegar a la zona del STOP. Si hubiera saturación en esa zona, habría un gran atasco. Si no es el caso, pero hay vehículos esperando en el STOP, tendrá que esperar tras ellos.
Si no hay vehículos parados en el STOP, comprobará si puede avanzar hacia la zona del semáforo, mirando si hay hueco en esa zona o, por el contrario, la cola del semáforo está llena de vehículos. Si está llena, no avanzará, sino que se quedará esperando en el STOP. Si hay hueco, tras simular una breve frenada en el STOP, avanza y comprueba el estado del semáforo. Si está en rojo, tendrá que esperar en el semáforo. Si no, podrá pasar sin problemas y llegar al aparcamiento.
La breve frenada se simulará llamando a la función vehi_frena, cuyo prototipo (en fase de diseño TAD, pseudocódigo) es: **`vehi_frena (Vehiculo pv);`**

Se ha decidido implementar la solución usando las siguientes estructuras y tipos de datos (definidas en los ficheros .c y .h correspondientes):

```
typedef enum {ERROR, ATASCAZO, EN_STOP, EN_SEM, EN_PARKING} Respuesta;
typedef enum {ROJO, AMBAR, VERDE} SemEstado;

struct _Semaforo {
    int num;    // id del semáforo, que también es el nº del semáforo en el mapa (0, 1, 2...)
    SemEstado estado;  // VERDE, AMBAR o ROJO
    Cola *colaVehic;   // cola de vehículos esperando en el semáforo
};
```

```
struct _STOP {
    int num;              // número identificador del STOP en el mapa (0, 1, 2...)
    Cola *colaVehic;      // cola de vehículos esperando en el STOP
};
struct _Mapa {
    Sem *sems [MAX_SEM];     //el num de un semáforo es el índice del semáforo en este array
                             //(el semáforo con num 0 ocupa la posición 0 de este array, etc)
    STOP *stops [MAX_STOPS]; //el num de un STOP es el índice del STOP en este array
    int numSems;             //nº de semáforos en el mapa
    int numSTOPs;            //nº de STOPs en el mapa
};
typedef struct  _Semaforo  Sem;
typedef struct  _STOP  STOP;
typedef struct _Mapa  Mapa;
typedef struct _Vehiculos  Vehic;
```

Con respecto a las funciones (primitivas o derivadas), en la fase de diseño de los TADs (previa a la implementación), se han diseñado las siguientes:

Bool cola_vacia (Cola pcola) Devuelve TRUE si la cola está vacía y FALSE si no.
Bool cola_llena (Cola pcola) Devuelve TRUE si la cola está llena y FALSE si no.

Status cola_creaEInserta (Cola pcola, Vehiculo pvehi) recibe una cola y un vehículo, y convierte el vehículo en un elemento de cola y lo inserta en la cola, devolviendo OK si lo inserta bien y ERROR si algún argumento de entrada es erróneo o si la cola está llena.

STOP m_getSTOP (Mapa mapa, entero i) Devuelve el STOP i-ésimo del mapa (sin hacer copia) o NULL

Sem m_getSem (Mapa mapa, entero i) Devuelve el semáforo i-ésimo del mapa (sin hacer copia) o NULL

Cola stop_getCola (STOP pstop); Obtiene la cola del stop pstop (no hace copia de la cola, la devuelve directamente)
Cola sem_getCola (Sem psem); Obtiene la cola del semáforo psem (no hace copia de la cola, la devuelve directamente)

Proporciona tú el **pseudocódigo, sin control de errores,** de la función de prototipo

Respuesta avanzaVehiculo (Mapa mapa, Vehiculo vehi);

que simula el avance de un vehículo por la vía de servicio, desde la UAM al instituto. La función devolverá:

- ATASCAZO: si el vehículo no logra ni entrar en la zona del STOP, porque está llena.
- EN_STOP: si el vehículo ha avanzado hasta la zona del STOP, pero se ha quedado esperando sin pasarlo porque había vehículos delante.
- EN_SEM: si el vehículo ha logrado pasar el STOPs sin esperar (solo simulando la frenada), pero se ha encontrado el semáforo en rojo y se ha tenido que quedar esperando a que el semáforo cambie su estado.
- EN_PARKING: si el vehículo no ha tenido que esperar en ningún sitio y ha logrado llegar al aparcamiento directamente (solo simulando la frenada).
- (ignorar) ERROR si se produce algún error en las llamadas a funciones o en los argumentos de entrada (no se puede simular hasta dónde llegaría el vehículo, se tendrá que volver a llamar a esta función para intentarlo de nuevo).

Nota: para indicar el número del stop o del semáforo del mapa que se desea mirar, se pueden usar directamente los números 0, 1,...

Nota2: también está disponible la función:

```
SemEstado   sem_getEstado  (Sem *ps);      /* Devuelve el estado del semáforo */
```

SOLUCION: Sin control de errores (tal y como se pedía)

OPCIÓN 1)

```
Respuesta mapa_avanzaVehículo (Mapa pmapa, Vehículo pvehi)

    pstop = m_getSTOP (pmapa, 0); //Obtiene el STOP, que es el primero de su zona
    pcola1 = stop_getCola (pstop); //Obtiene la cola del stop

    //Si la cola en el STOP está llena, ATASCAZO
    si cola_llena (pcola1) = TRUE:
        devolver ATASCAZO;

    //Si hay cola en el STOP, se encola para esperar tras los vehículos que ya están allí
    si cola_vacia(pcola1) = FALSE:
        cola_creaEInserta (pcola1, pvehi);
        devolver EN_STOP;

    //Si no hay nadie en el STOP, mira si puede seguir hacia la zona del semáforo

    psem = m_getSem (pmapa, 0); //Obtiene el primer semáforo de la zona
    pcola2 = sem_getCola (psem);  //Obtiene la cola del semáforo (en pcola1 mantengo la cola del stop)

    //Si la cola del semáforo está llena, debe quedarse en el STOP (será el 1er vehículo en esa cola)
    si cola_llena (pcola2) = TRUE:
        cola_creaEInserta (pcola1, pvehi);
        devolver EN_STOP;

    //Si no, simula frenada y avanza hacia la zona del semáforo.
    vehi_frena (pc);

    //Cuando llega a la zona del semáforo, si está rojo se encola y si no pasa directo al aparcamiento
    //tal y como indica el enunciado

    si sem_getEstado (psem) = ROJO:
        cola_creaEInserta (pcola2, pvehi); //pcola2 tiene la cola del semáforo
        devolver EN_SEM;

    devolver EN_PARKING;  //si no se ha quedado en ninguna cola, ha pasado hasta el parking
```

OPCIÓN 2) Misma solución, pero con el código de encolar en la cola del STOP una sola vez, obteniendo todas las colas al principio, incluso aunque puede que no se use la del semáforo:

```
Respuesta mapa_avanzaVehículo (Mapa pmapa, Vehículo pvehi){
    pstop = m_getSTOP (pmapa, 0);  //Obtiene el STOP, que es el primero de su zona
    pcola1 = stop_getCola (pstop); //Obtiene la cola del stop
    psem = m_getSem (pmapa, 0);    //Obtiene el primer semáforo de la zona
    pcola2 = sem_getCola (psem);   //Obtiene la cola del semáforo

    //Si la cola en el STOP está llena, ATASCAZO
    si cola_llena (pcola1) = TRUE:
        devolver ATASCAZO;

    //Si hay cola en el STOP o la cola del semáforo está llena, se queda en el STOP
    si cola_vacia(pcola1) = FALSE o cola_llena (pcola2) = TRUE:
        cola_creaEInserta (pcola1, pvehi);
        devolver EN_STOP;

    //Si no, simula frenada y avanza hacia la zona del semáforo.
    vehi_frena (pc);

    //Cuando llega a la zona del semáforo, si está rojo se encola y si no pasa directo al aparcamiento
    si sem_getEstado (psem) = ROJO:
        cola_creaEInserta (pcola2, pvehi); //pcola2 tiene la cola del semáforo
        devolver EN_SEM;

    devolver EN_PARKING;
```

OTRAS OPCIONES CORRECTAS:

- Encolarse en las colas y salir de ellas solo si se puede (menos eficiente, pero correcta desde el punto de vista lógico):
    - Al principio se mete en la cola del stop, haya coches o no. Si la del semáforo no está llena, sale de la cola del stop y se mete en la del semáforo.
    - Si, una vez que está en la cola del semáforo, ve que el semáforo está verde, sale de la cola del semáforo para seguir hacia el parking.
- También es correcto simular la frenada (vehi_frena(pc);) no solo antes de pasar el STOP, sino también cuando se mete en cualquiera de las colas.
- Correcto si en vez de la función sem_getEstado se ha utilizado otra inventada con otro nombre.