1. En un cumpleaños de niños pequeños se pretende repartir trozos de tarta a todos los niños. Como un día especial que es, los niños pueden repetir tantas veces como quieran y cuando quieran pero usando un sistema de rondas, de tal forma que hasta que todos los niños no han recibido su trozo de tarta de una ronda, los niños que ya han tomado tarta no pueden repetir.

   Implementa el proceso `niñoPideTarta(int NumNiño)` teniendo en cuenta que los semáforos que se van a utilizar son semáforos blandos (o también denominados débiles). Se dispone de la función auxiliar `comeTarta()` para representar la tarea de comer el trozo de tarta. De igual forma, se utilizarán las funciones `up(semaforo)` y `down(semaforo)` para manipular los semáforos. Se pueden crear otras funciones si se consideran necesarias.

   Solución:

```
semaforo semNiño[NUMERONIÑOS]={1,1,...,1}
semaforo mutex=1;
int cuenta = 0;

niñoPideTarta(int i)
{
    int j;

    while (TRUE) { // mientras que dure la fiesta
        down(semNiño[i]);
        down(mutex);
          ++cuenta;
          if (cuenta == NUMERONIÑOS)
          {
            cuenta = 0;
            for(j=0; j < NUMERONIÑOS; ++j)
               up(semNiño[j]);
          }
        up(mutex);
        comeTarta();
    }
}
```

2. Santa Klaus es un hombre muy mayor y necesita dormir mucho; por ello aprovecha siempre que puede para dormir.

   Los elfos saben de su necesidad y han decidido que aunque haya un problema en la fábrica de juguetes no le despertarán, pero cuando haya tres problemas entonces sí que lo harán.

   La fábrica y la oficina de Santa están muy alejadas, por lo que si un elfo va en busca de Santa no vuelve hasta que Santa resuelva su problema.

   Los elfos irán a despertar a Santa pero si no hay otros dos elfos esperando es que no hay suficientes problemas como para despertar a Santa, por ello y ya que no tienen nada que hacer se quedarán a la espera de que Santa los despierte. El tercer elfo hará lo mismo pero despertará a Santa. Cuando Santa resuelva los problemas en la fábrica despertará a los tres elfos.

   Así mismo los renos están de vacaciones todo el año y vuelven por navidad para tirar del trineo de Santa. Cuando llegue el último de los 9 renos despertará a Santa para irse a repartir los juguetes.

   Condiciones:

   - No es necesario controlar lo que sucede con los elfos y la producción cuando Santa está de reparto.
   - Utiliza las primitivas **up** y **down**.

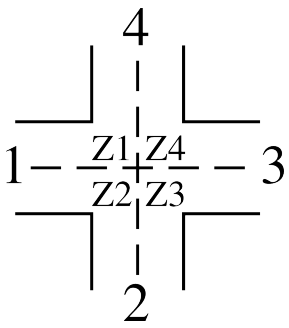
Se pide: crear los procesos **Santa**, **elfo** y **reno** y definir las variables compartidas y los semáforos con sus valores iniciales. No es necesario crear ningún programa que lance los procesos.

3. En la Escuela Superior de Hacking todos los aseos son *unisex*. Para usar estos aseos hay un selector a la entrada que tiene tres posiciones: hombres, mujeres y libre. El cartel de hombres lo pone el primer hombre que entre, el de mujeres la primera mujer que entre y el de libre lo pone cualquiera al salir siempre que quede vacío el aseo. Cuando está puesto el cartel de mujeres sólo pueden entrar mujeres y cuando pone el cartel de hombres sólo pueden entrar hombres. En el aseo sólo caben 10 personas por lo que cuando está lleno se activará un cartel que indica que está lleno y los que lleguen esperarán fuera a que haya sitio.

   Resuelve el problema implementando el pseudocódigo de los procesos `mujer()` y de `hombre()` para que cumplan las condiciones. No es necesario prevenir la inanición.

4. En el año 2056 se realiza una modificación en el código de la circulación en la que cambia la regulación de los cruces con bajo tráfico. Dada la capacidad de las comunicaciones inalámbricas cuando un coche se acerca a un cruce reserva los elementos necesarios del cruce para poder realizar la maniobra.

   Si no puede acceder a una de las zonas necesarias para realizar la maniobra no accederá al cruce. Es muy importante que si las maniobras de varios coches no interfieren todos los coches puedan acceder al cruce sin ningún problema. Crea el proceso `coche(origen,destino)` donde origen y destino determinan las zonas a utilizar. En el esquema adjunto se puede ver la numeración de zonas así como la numeración de origen/destino. Por ejemplo, si el origen es 2 y el destino es 1 es necesario reservar las zonas Z3, Z4, Z1, sin embargo de 1 a 2 hace falta reservar sólo Z2. Utiliza las primitivas `up(sem)`, `down(sem)`, `up(sem,n)`, `down(sem,n)` (las dos últimas para hacer ups y downs múltiples si es necesario). No es necesario prevenir inanición pero sí evitar interbloqueos.



Solución:

Se define un "array" de semáforos todos inicializados a 1 en el que el primer elemento se utilizará como semáforo mutex para bloquear la reserva de recursos (elementos del cruce).

```
semaforo Z[5]={1,1,1,1,1}

choche(origen, destino)
{
    if(destino < origen) destino += 4;

    down(Z[0]);
    for(i=origen; i < destino; ++i) down(Z[i%4+1]);
    up(Z[0]);

    cruza();

    for(i=origen; i < destino; ++i) up(Z[i%4+1]);
}
```

5. En una fábrica de coches hay 5 cadenas de montaje. De cada una de ellas salen los coches de uno en uno. Estos coches son almacenados en una zona de espera. En esta zona de espera hay 3 dársenas donde paran los camiones de transporte que pueden transportar cada uno 6 coches. Los camiones nunca parten antes de estar completamente cargados. Cuando un camión desea cargar puede hacerlo siempre y cuando uno que estuviese cargando pueda terminar la carga.

   Implementa los procesos `cadenaMontaje y cargaCamion`. Utiliza las primitivas `up(sem)`, `down(sem)`, `up(sem,n)`, `down(sem,n)` (las dos últimas para hacer ups y downs múltiples si es necesario).

No es necesario controlar las posiciones donde se encuentran los coches a cargar ni implementar el acceso a las dársenas ni prevenir inanición.

6. La basílica de la Sagrada Familia en Barcelona dispone de un museo en el que se explican los métodos de construcción y maquetas de prueba de carga. los responsables de la visita de turistas a la basílica están hartos de que los turistas entren a ver la basílica sin saber lo que están viendo y han decidido que es necesario visitar el museo antes de entrar en la basílica. Además, han decidido que la basílica sólo se pueda visitar con un guía que explique lo que estamos viendo. Nos piden implementar un sistema que permita realizar esto de forma eficiente y con las siguientes condiciones:

   - El número máximo de personas en el museo es de 100 personas.
   - Cuando un visitante abandona el museo se pone en la cola de espera para que los recoja un guía.
   - Hay 5 guías.
   - Un guía recoge a grupos de 10 personas.
   - Si no hay 10 personas en la cola de espera el guía espera a que las haya y si hay más sólo coge a 10.

   Implementa el sistema usando semáforos.

Solución:

```
int cuenta = 0;
semaforo museo = 100, mutex = 1, visita = 0, grupo = 0, guia = 5;

visitante()
{
    down(museo);
    visitaMuseo();
    up(museo); /* Al final si es un total de 100 */

    down(mutex);
    ++cuenta;
    if(cuenta == 10)
    {
        cuenta = 0;
        up(visita);
    }
    up(mutex);

    down(grupo);
}

guía()
{
    while(1)
    {
        down(guia);
        down(visita);
        up(grupo,10);
        visitaBasilica();
        up(guia);
    }
}
```

7. Umeå es una pequeña ciudad sueca en el círculo polar ártico. Debido a las inclemencias climáticas su principal medio de transporte es un pequeño avión con un máximo de 32 plazas para viajeros. El pequeño aeropuerto tiene una única terminal de embarque. Con el fin de obtener rentabilidad en la línea de comunicación con Estocolmo y reducir costes, la línea aérea se usa de forma muy similar a la de una línea de autobuses pero por seguridad con unas pocas reglas que es necesario cumplir:

    1. Cuando el avión está preparado, éste se llena con los pasajeros que estén esperando. Si hay más de 32 pasajeros esperando se llenará con 32 sin un orden determinado. Los pasajeros que no hayan podido embarcar esperarán al siguiente avión.
    2. Si no hubiera pasajeros esperando, el avión volverá de forma inmediata a Estocolmo.
    3. Si los pasajeros están embarcando nadie que llegue después de que se inicie el embarque podrá entrar en la zona de embarque y mucho menos embarcar (independientemente de si los pasajeros llenan el avión o no) y sólo podrán entrar en la zona de embarque tras la partida del avión.

8. En un quirófano se va a realizar una apendicectomía. Para ello actúan tres profesionales sanitarios: un anestesista, una enfermera y un cirujano. Las tareas de cada uno son:

    - El anestesista
        - Inicia todo el proceso anestesiando al paciente.
        - No permite que la enfermera empiece a actuar hasta que el paciente no esté sedado.
        - Mantiene la correcta sedación.
        - Cuando la enfermera lo indique puede eliminar la sedación y empezar el "despertar".
    - La enfermera
        - Prepara el instrumental y las gasas.
        - No permite que el cirujano empiece a actuar mientras no se haya comprobado todo el instrumental y las gasas.
        - Aporta el material que solicita el cirujano contando el instrumental y las gasas que se utilizan y los que se dejan de utilizar.
        - No permite que el cirujano cosa si no se ha devuelto todo el material aportado al cirujano (porque si no el pobre paciente se quedaría con algo dentro).
    - El cirujano
        - Empieza a operar y pide 3 instrumentos y dos gasas.
        - Devuelve 2 instrumentos y 1 gasa.
        - Pide 1 instrumento y una gasa.
        - Quiere coser
        - La enfermera le indica el instrumental a devolver
        - Lo devuelve
        - Cose

    Crea los procesos necesarios y sincronízalos adecuadamente usando variables compartidas y semáforos indicando sus valores de inicialización.

9. En el ayuntamiento de Navalaoveja se han gastado todo el presupuesto en mármol. Como no tienen presupuesto suficiente han despedido a policías municipales pero aún así tienen que mantener la seguridad en los pasos de cebra a la salida de los colegios. Como no hay presupuesto suficiente nos piden que implementemos un sistema basado en semáforos para que tanto los peatones como los automóviles cumplan con las condiciones siguientes sin necesidad de policías:

    - Si no hay automóvil que quieran acceder al paso de cebra los peatones deben poder cruzar cuando lo deseen.
    - Si no hay peatones que quieran acceder al paso de cebra los automóviles deben poder pasar cuando lo deseen.

- Si hay diez automóviles esperando a que pasen peatones deben dejar de cruzar peatones y pasar los 10 coches. Aunque haya más coches, sólo pasarán 10 si hay peatones esperando.
- Salvo en el caso anterior los peatones siempre tienen prioridad para cruzar.
- Como el uso del paso de cebra en ambas direcciones tanto de peatones como de automóviles no es exclusivo, no se tendrá en cuenta la dirección en la que desean pasar.
- Evidentemente los peatones pueden pasar en grupos pero los coches sólo pueden pasar de uno en uno.

Implementa los procesos `peaton()`, `coche()` y `policia()` usando semáforos. Para ello puedes usar las funciones `up(sem)`, `down(sem)`, `up(sem,cant)` y `down(sem,cant)`. También puedes usar las funciones `llegapeaton()`, `llegacoche()`, `cruzapeaton()`, `cruzacoche()`, `continuapeaton()` y `continuacoche()`.

Solución:

```
#define MAXAUTO 10

semaforo esperapeaton=1, mutexpeaton=0;
semaforo paso=1;
semaforo esperacoche=1, mutexcoche=0;
int cuentacoche=0, cuentapeaton=0;

peaton()
{
  llegapeaton();

  down(esperapeaton);
  up(esperapaton);

  down(mutexpeaton);
    ++cuentapeaton;
    if(cuentapeaton == 1) down (paso);
  up(mutexpeaton);

  cruzapeaton();

  down(mutexpeaton);
    --cuentapeaton;
    if(cuentapeaton == 0) up (paso);
  up(mutexpeaton);

  continuapeaton();
}
```

```
coche()
{
  llegacoche();

  down(esperacoche);
  up(esperacoche);

  down(mutexcoche);
    ++cuentacoches;
    if(cuentacoches == MAXAUTO)
    {
      down(esperacoche);
      down(esperapaton);
    }
  up(mutexcoche);

  down(paso);
  cruzacoche();
  up(paso);

  down(mutexcoche);
    --cuentacoches;
    if (cuentacoches == 0)
    {
      up(esperapeaton);
      up(esperacoche);
    }
  up(mutexcoche);

  continuacoche();
}
```

10. La comisaría de policía de Ciudad Lineal está especializada en gestiones relacionadas con los DNIs o pasaportes de ciudadanos. El acceso al recinto está regulado por un agente de policía que estudia filosofía y se pasa tiempo pensando en sus cosas de los estudios.

    Los ciudadanos esperan en la puerta de la comisaria en cola por orden de llegada. Como el agente cuando se pone a pensar no se da cuenta del tiempo que ha transcurrido, se ha puesto un programa en el móvil que

le avisa cada 15 min. Después de cada aviso, el agente actúa del siguiente modo: Si en la cola hay 10 o más ciudadanos da paso a los 10 primeros ciudadanos que realizarán sus gestiones en la comisaría y el agente vuelve a ponerse a pensar en sus cosas de filosofía. Si no hay ciudadanos en la cola el agente vuelve a pensar en sus cosas de filosofía. Si hay menos de 10 ciudadanos, el agente da paso a todos los ciudadanos que están esperando y se pone a pensar en sus cosas de filosofía.

Escribir el pseudocódigo de un programa que usando sólo semáforos binarios coordine la actividad del agente y los ciudadanos para acceder a la comisaría. El pseudocódigo del programa debe incorporar la declaración de variables, código del proceso ciudadano `void ciudadano( )`, código del proceso agente void agente( ) y el programa principal para lanzar la ejecución concurrente de los procesos. El tiempo de llegada de cada ciudadano a la cola es aleatorio.

Se dispone de las funciones:

- **init_sem (sem, valor)** que inicializa el semáforo sem con el valor valor
- **realizar_gestion ()**, función para que el ciudadano pueda realizar su gestión
- **alarm (tiempo)**, función para avisar después de tiempo segundos.
- **pensar ()**, función en la que se bloquea a la espera del aviso de su móvil.

11. El responsable de la cafetería de la EPS nos ha pedido que regulemos con semáforos el acceso de los estudiantes a las mesas del comedor. Los estudiantes deben seguir el siguiente protocolo:

    - Cuando un estudiante llega, lo que hace es buscar una silla libre. Si no encuentra sitio, se va sin esperar. Si localiza una silla libre, la reserva poniendo su mochila en la silla.
    - Una vez que ha reservado la silla, se va a buscar una bandeja. Si no hay bandejas libres, esperará haciendo cola hasta que se liberen bandejas. Si hay bandejas libres, se pondrá a la cola para que el cocinero le sirva la comida.
    - Una vez que le sirven la comida se sienta en la silla y cuando termina de comer se levanta de la silla, deja la bandeja en el portabandejas y se va.

    El número de sillas del comedor es S y el número de bandejas B. Resuelva este problema mediante semáforos, indicando los que necesitamos (nombre y estado inicial), y escribiendo el pseudocódigo de los procesos `estudiante()` y `cocinero( )`.

12. La comunidad de Castilla y León ha tomado la decisión de que el aeropuerto de Villanubla (Valladolid) se utilice como base de operaciones de una flota de aviones. Los aviones tienen que utilizar de forma ordenada la pista de tal forma que ningún avión puede utilizar la pista mientras otro la esté utilizando para aterrizar o despegar. Así mismo los aviones que quieren aterrizar tienen preferencia ante los que quieren despegar porque están consumiendo combustible. Entre los que quieren aterrizar hay dos tipos de aviones, los que tienen poco combustible y los que no. Los que tienen poco combustible tendrán preferencia frente al resto de los aviones que quieren aterrizar.

    Utiliza las primitivas `up()` y `down()` para crear los procesos de los tres tipos de aviones.

Solución: Esquema:

- aterrizaUrgente()–Lector1-2
- aterriza–Escritor1 y dentro Lector2
- despega–Escritor2

```
int cuenta = 0, cuentaUr = 0;

semaforo m1 = 1, m2 = 1, pista = 1, aterrizaje = 1;
```

```
aterriza_Urgente()
{
   down(m1);
   cuentaUr ++;
   if (cuentaUr == 1)
   {
      down(aterrizaje);
      down(pista);
   }
   up(m1);

   AterrizaComoPuedas()

   down(m1);
   cuentaUr --;
   if (cuentaUr == 0)
   {
      up(pista);
      up(aterrizaje);
   }
   up(m1);
}
```

```
aterriza_Standard()
{
   down(aterrizaje);

   down(m2);
   cuenta ++;
   if (cuenta == 1) down(pista);
   up(m2);

   AterrizaComoPuedas()

   down(m2);
   cuenta --;
   if (cuenta == 0) up(pista);
   up(m2);

   up(aterrizaje);
}
despega()
{
   /* deja paso */
   down(aterrizaje);
   up(aterrizaje);
   up(aterrizaje);
   down(aterrizaje);

   /* usa pista */
   down(pista);
   despegar();
   up(pista);
}
```

13. En una fábrica un conjunto de operarios llenan cajas de fresas de 2 Kg. cada una y según las llenan las van dejando en una superficie en la que caben 100 de esas cajas. Cuando hay al menos 10 cajas de fresas uno de los empaquetadores junta 10 en un paquete de transporte y lo guarda en el almacén. En este almacén se pueden almacenar hasta 100 de estos paquetes. Estos paquetes se guardan en camiones. Según llegan los camiones se van llenando hasta que llenan el camión con 10 paquetes en cuyo caso se marchan. Eso sí, no se puede meter ningún paquete en ningún camión si hay alguno que está siendo cargado. Determina qué procesos hay que sincronizar, qué semáforos es necesario utilizar y qué valor inicial tienen e implementa los procesos en forma de pseudocódigo. Utiliza las funciones que consideres necesario.

14. En una carrera de fórmula 1 el equipo de competición está formado por el piloto y un número importante de mecánicos especializados en hacer una rutina muy concreta.

    Los mecánicos son: mecánico de parada, mecánico de levantada, 4 mecánicos de tuercas, 4 mecánicos de quitar ruedas y 4 mecánicos de poner ruedas. En total 15 mecánicos.

    Cuando un coche entra en *boxes* para hacer un *pit stop*, se sigue el siguiente protocolo:

    - El mecánico de parada comprueba que el resto de mecánicos están disponibles.
    - El piloto para y espera.
    - El mecánico "parada" activa la señal de coche parado.
    - Cuando esté activada la señal de coche parado el mecánico de "levantada" levanta el coche con un gato hidráulico.
    - Cuando el coche esté levantado los mecánicos de "tuerca" quitan las tuercas de cada rueda (4 mecánicos de tuerca, uno por cada rueda).
    - Cuando se han quitado las tuercas los mecánicos que "quitan" ruedas actuan (4 mecánicos, uno por rueda).

- Cuando se han quitado las ruedas los mecánicos que "ponen" ruedas lo hacen (4 mecánicos, uno por rueda).
- Cuando se han puesto las ruedas los mecánicos de "tuerca" colocan las tuercas (los mismos mecánicos de tuerca de antes).
- Cuando se han puesto todas las tuercas el mecánico de "levantada" baja el coche.
- Cuando el coche ha sido bajado el mecánico de "parada" activa la señal de marcha.
- Finalmente, el piloto acelera y continua con la carrera.

Crea un programa que lance de forma concurrente los procesos que sean necesarios (`piloto()`, `m_parada()`, `m_levantada()`, `m_tuerca()`, `m_quitan()` y `m_ponen()`).

Indica qué variables compartidas necesitas y en cada una de ella cuál es su valor inicial.

Solución:

```
#define RUEDAS 4
#define PILOTO 1
#define STOP 1
#define LEVANTAR 1

semaforos prepMecanicos = prepParada = llegadaCoche= salidaCoche= MLevantada = 0;
semáforos finLevantada = MTuercas = tuercaQuitada = MQuitaRueda = ruedaQuitada =0;
semáforos MPoneRueda= ruedaPuesta= MTuercasB= tuercaPuesta =0;

/*
 * Main
 */

main()
{
    int i;
    int equipo=PILOTO+STOP+LEVANTAR+3*RUEDAS;

    /*
      * Lanzamiento Procesos
      */

    if(!fork()) { piloto(); }
    if(!fork()) {m_parada(); }
    if(!fork()) {m_levantada(); }
    for(i=0; i<RUEDAS; ++i)
           if(!fork()) {m_tuerca();}
    for(i=0; i<RUEDAS; ++i)
           if(!fork()) {m_quitan();}
    for(i=0; i<RUEDAS; ++i)
           if(!fork()) {m_ponen();}

    /*
      * Espera por todos los procesos
     */
```

```
    for(i=0; i<equipo; ++i)
           wait();

    exit(EXIT_SUCCESS);

} // end main()

/*
 * Piloto
 */

    void piloto()
    {

        down(prepParada); // Espera a que el mecánico de Parada le dé paso
        paracoche();
        up(llegadaCoche); // Avisa al mecánico de Parada de que ya está el coche parado
        down(salidaCoche); //Espera a que el mecánico de Parada le avise de que puede salir
        acelera();
        exit(EXIT_SUCESS);

  } // End piloto()

/*
 * Mecánico Parada
 */

    void m_parada()
    {

        down(prepMecanicos, 3*RUEDAS + 1); // Espera a que estén todos los mecánicos preparados
        up(prepParada); // Permite que entre el coche a cambiar las ruedas
        down(llegadaCoche); // Espera a que llegue el coche
        up(MLevantada); // Permite que el mecánico de levantada levante el coche
        down(finLevantada); // Espera a que el mecánico de levantada baje el gato
        up(salidaCoche); // Permite que salga el coche
        exit(EXIT_SUCESS);

    } //End mecanico_parada

/*
 * Mecánico Levantada
 */

m_levantada()
{

    up(prepMecanicos); // Envía la señal de disponible al mecánico de Parada
    down(MLevantada); // Espera indicación del mecánico de Parada, coche en la plataforma
    sube_coche( );
    up(Mtuercas,4); // Avisa a los mecánicos de tuercas
    down(tuercaQuitada,4); // Espera aviso de los mecánicos de tuercas, tuercas quitadas
```

```
    up(MQuitaRueda,4); // Avisa a los mecánicos de quitar ruedas
    down(ruedaQuitada,4); //Espera a que las ruedas estén quitadas
    up(MPoneRueda,4); // Avisa a los mecánicos de poner ruedas
    down(ruedaPuesta,4); //Espera a que las ruedas estén colocadas
    up(MtuercasB,4); // Avisa a los mecánicos de tuercas que pongan las tuercas
    down(tuercaPuesta,4); // Espera a que las tuercas estén puestas
    bajaCoche();
    up(finLevantada); //Avisa al mecánico de Parada de que el coche está listo
    exit(EXIT_SUCESS);

} // End mecanico levantada

/*
 * Mecánico Tuerca
 */

m_tuercas()
{

    up(prepMecanicos); // Envía la señal de disponible al mecánico de Parada
    down(Mtuercas); //Espera la orden de comienzo
    quitaTuerca();
    up(tuercasQuitada); //Avisa de que ha terminado de quitar su tuerca
    down(MtuercasB); // Espera a que le avisen
    poneTuerca();
    up(tuercaPuesta); /Avisa de que ya está la tuerca puesta
    exit(EXIT_SUCESS);

} // End mecanico tuercas

/*
 * Mecánico Quita-Ruedas
 */

m_quitan()
{

    up(prepMecanicos); // Envía la señal de disponible al mecánico de Parada
    down(MQuitaRueda); //Espera el aviso para quitar la rueda
    quitaRueda();
    up(ruedaQuitada);
    exit(EXIT_SUCESS);

} // End mecanico quitar ruedas

/*
 * Mecánico Pone-Ruedas
 */

m_ponen()
{
```

```
    up(prepMecanicos); // Envía la señal de disponible al mecánico de Parada
    down(MPoneRueda); //Espera el aviso para quitar la rueda
    poneRueda();
    up(ruedaPuesta); //Avisa de que las ruedas están colocadas
    exit(EXIT_SUCESS);

} // End mecánico poner ruedas
```

15. En Coria del Río, en las marismas del Guadalquivir hay un sistema trasbordador que, de usarlo, permite ahorrar 30 km en automóvil con sólo 300 m de trasbordaje por el río. En este sistema, hay dos barcazas que admiten 4 automóviles cada una y varias personas. Dado los bajos precios que son necesarios para que sea un servicio bastante usado, los propietarios nos piden implementar un sistema que permite maximizar el rendimiento minimizando los tiempos de espera. Para ello nos piden un sistema con las siguientes condiciones:

    - Una barcaza no sale hasta que no esté llena de automóviles.
    - Una barcaza no admite automóviles si la otra barcaza tiene algún automóvil y aún no ha salido.
    - Las barcazas pueden estar o no al mismo tiempo en el embarcadero.
    - Sólo hace falta implementar uno de los embarcaderos porque se supone que el otro es idéntico en comportamiento.
    - Los automóviles suben en la barcaza que tenga abiertas sus puertas y si las tienen abiertas las dos barcazas los automóviles suben en una de ellas aleatoriamente (son introducidos por personal de la empresa).

    Implementa los procesos `automovil()` y `barcaza()` usando semáforos. Para ello puedes usar las funciones `up(sem)`, `down(sem)`, `up(sem,cant)` y `down(sem,cant)`.

    Solución:

```
#define MAXAUTO 4

semaforo esperando=0, auto=0;
semaforo embarca=1, puertas=1;
int cuentaPers=0, cuentaAuto=0;

void automovil()
{
    llegaAuto();
    down(auto);
    down(embarca);
    embarcando();
    ++cuentaAuto;
    if(cuentaAuto == 4)
    {
            cuentaAuto = 0;
            up(esperando);
    }
    up(embarca);
}
```

```
void barcaza()
{
    llegaBarcaza();
    down(puertas);
    abrePuerta();
    up(auto,4);
    down(esperando);
    up(puertas);
    saleBarcaza();
}
```

16. Para resolver las necesidades de aulas derivadas de la implantación de los nuevos planes de estudio, la Escuela Monotécnica Inferior del campus de Piedranegra ha decidido convertir la mitad de los servicios (baños) en

aulas. Eso crea un problema con los servicios, los cuales deberán ser compartidos por chicos y chicas. Para evitar situaciones inconvenientes y políticamente incorrectas, se ha implementado un protocolo para asegurar que en un momento dado, sólo haya chicos o sólo haya chicas dentro del servicio. En particular, un chico sólo puede entrar si en el servicio no hay nadie o sólo hay chicos; de la misma forma, una chica sólo puede entrar si no hay nadie o hay sólo chicas. Se pide implementar con semáforos el protocolo de los procesos `Chica( )` y `Chico( )`.

17. Razona cuántos semáforos se necesitan si:

    1. Dos procesos P1 y P2 tienen que sincronizarse en un punto.
    2. Tres procesos P1, P2 y P3 tienen que sincronizarse según la siguiente secuencia temporal: P1 con P2, P2 con P3 y por último P1 con P3.

    Nota: Todos los procesos se ejecutan una sola vez y no contienen bucles.

18. Considera las siguientes relaciones de precedencia entre procesos:

    - P1 antes de P2 y P3
    - P2 antes de P4 y P5
    - P3 antes de P5
    - P6 después de P3 y P4

    donde Pi antes de Pj significa que la ejecución de Pi debe ser completada antes de que la ejecución de Pj comience y Pi después de Pj significa que Pi debe comenzar a ejecutarse después de Pj haya terminado. Declara, inicializa y utiliza los semáforos necesarios para que cada proceso sea forzado a ejecutarse según las relaciones de precedencia establecidas anteriormente.

```
Proceso P1( )                Proceso P2( )                          Proceso P6( )
{                            {                                      {
   .....                        .....                                  .....
   Código P1                    Código P2                  ...         Código P6
   ....                         ....                                   ....

   exit(EXIT_SUCCCESS);         exit(EXIT_SUCCCESS);                   exit(EXIT_SUCCCESS);
}                            }                                      }
```

19. El Bosque de los Arrayanes en Bariloche, Argentina, es una formación natural única en el mundo, que recibe anualmente la visita de miles de turistas. Se trata del único bosque existente en el mundo de este tipo de arbusto, lo que unido a su singular belleza, da lugar a un especial atractivo. Esto provoca que muchos turistas se quieran llevar "un recuerdo", arrancando muchos de ellos ramas de los arbustos.

    Para evitar que los turistas deterioren el bosque, no se permiten las visitas si no son acompañadas por un guardia forestal y los grupos nunca deben ser mayores de 20 personas. Así, las visitas se inician cuando se forma un grupo de 20 personas o cada 15 minutos, si hay visitantes esperando pero no han conseguido formar un grupo completo.

    Los guardias forestales prefieren esperar indefinidamente hasta que se alcanza el número máximo de personas permitidas en un grupo y de esa forma trabajar menos. Para evitar que hagan esperar demasiado a los turistas, la dirección ha colocado en la entrada del bosque un inspector que se encarga de vigilar que ningún guardia forestal permanezca más de 15 minutos esperando a que se forme su grupo, ordenándoles partir con los turistas que estén esperando, si hubiera alguno. Si no hubiera turistas, se inicia un nuevo periodo de 15 minutos de espera. Para facilitar la tarea de control del inspector, los guardias deben esperar a que les llegue el turno de formar un grupo de turistas.

    Se pide modelar con semáforos la situación descrita, mostrando el pseudo-código correspondiente a los procesos `Guardia_Forestal()`, `Turista()` e `Inspector()`, respectivamente.

    Se recuerda que los semáforos a utilizar sólo pueden ser accedidos mediante las funciones vistas en clase de teoría, `down(sem)` y `up(sem)` y variables compartidas, si fuera preciso.

Para todas las variables que se utilicen, se debe especificar tipo, su valor de inicialización y si son compartidas entre varios procesos o no.

Se debe dar una lista de semáforos utilizados, indicando, para cada uno de ellos, qué proceso(s) ejecuta(n) el/los wait(s) y cuál(es) el/los signal(s) correspondientes, y en qué momento.

20. En el parque natural de Cabo de Gata hay un antiguo puente por el que sólo cabe un coche de ancho. Como es un parque natural las modificaciones que se pueden realizar son muy limitadas y por tanto el puente sólo se puede mantener, no se puede ensanchar ni reforzar. Como su estructura es muy débil sólo se permite que haya simultáneamente 10 vehículos. Como por el puente sólo cabe un coche de ancho, mientras haya vehículos circulando en una dirección no se permitirá que circulen vehículos en la contraria. Implementa un sistema que cumpla, por tanto, las siguientes condiciones:

    1. No permita más de cinco vehículos en el puente.
    2. Asegura que, una vez que un vehículo ha empezado a usar el puente, ningún otro vehículo intentará entrar en la dirección opuesta.
    3. No es necesario prevenir la inanición, es decir, una fila interminable de coches en una dirección puede provocar inanición sobre los coches que pretenden recorrer el puente en la otra dirección. Se supone que nunca existirá una fila interminable de coches. Existe una solución que previenen la inanición pero es muy ineficaz en el uso del recurso.

    Para ello es necesario crear un programa que lance aleatoriamente en el tiempo procesos `cocheVa()` y `cocheViene()`. También es necesario que los procesos `cocheVa()` y `cocheViene(int)` dispongan de la sincronización anteriormente indicada. Asumir la existencia de las funciones: `cruzarPuente()` y `nuevoCoche()`. La última de estas funciones espera un tiempo determinado y devuelve 1 o 0 dependiendo de la dirección de cruce del siguiente coche que va a aparecer. De igual forma, se utilizarán las funciones `up(semaforo)`, `up(semaforo,n)`, `down(semaforo)` y `down(semaforo,n)` para manipular los semáforos. Se pueden crear otras funciones si se consideran necesarias.

21. En el hospital El Descanso hay un pasillo elevado por el que se trasladan enfermos en camas. Es tan estrecho que sólo caben camas en una dirección. Como es un pasillo elevado, por seguridad no debe haber más de siete camas con enfermos trasladándose por el pasillo y como es tan estrecho no pueden circular camas en ambas direcciones simultáneamente. Implementa un sistema que cumpla, por tanto, las siguientes condiciones:

    1. No permita más de siete camas en el puente.
    2. Asegura que, una vez que una cama ha empezado a usar el pasillo, ninguna otra cama intentará entrar en la dirección opuesta.
    3. No es necesario prevenir la inanición, es decir, una fila interminable de camas en una dirección puede provocar inanición sobre las camas que pretenden recorrer el pasillo en la otra dirección. Se supone que nunca existirá una fila interminable de camas. Existe una solución que previenen la inanición pero es muy ineficaz en el uso del recurso.

    Para ello es necesario crear un programa que lance aleatoriamente en el tiempo procesos `camaVa()` y `camaViene()`. También es necesario que los procesos `camaVa()` y `camaViene(int)` dispongan de la sincronización anteriormente indicada. Asumir la existencia de las funciones: `cruzarPasillo()` y `nuevaCama()`. La última de estas funciones espera un tiempo determinado y devuelve 1 o 0 dependiendo de la dirección uso del pasillo de la siguiente cama que va a aparecer. De igual forma, se utilizarán las funciones `up(semaforo)`, `up(semaforo,n)`, `down(semaforo)` y `down(semaforo,n)` para manipular los semáforos. Se pueden crear otras funciones si se consideran necesarias.