**4.** (*2 p*) En un restaurante autoservicio inicialmente vacío existen S puestos para comer. Cuando un cliente llega al restaurante lo primero que hace es buscar un puesto libre, si no encuentra ninguno se marcha. Si encuentra algún puesto libre lo reserva dejando allí sus cosas. A continuación coge, sin necesidad de esperar ninguna cola, lo que desea comer de unos mostradores. Luego se pone en una cola para que le cobre un dependiente. Finalmente vuelve a su puesto para comer. El dependiente sólo atiende a los clientes de uno en uno y debe ser avisado por el cliente para que le cobre. Cuando no está cobrando a los clientes el dependiente se dedica a reponer los mostradores. Escribir en pseudocódigo un programa de nombre `autoservicio` que usando semáforos coordine la actividad de los clientes y del dependiente. Dicho programa debe tener cuatro partes: declaración de variables y semáforos, código del proceso `cliente`, código del proceso `dependiente` y código para inicializar los semáforos y lanzar la ejecución concurrente de ambos procesos.

**Solución:**

La solución que se propone para este problema tiene en cuenta lo siguiente:

- Cuando un cliente llega al restaurante, lo primero que hace es buscar un puesto libre, de los S disponibles, sino encuentra ninguno se marcha. Para modelar esta situación se va a utilizar una variable global de nombre `puestos_libres` que será consultada por los procesos cliente. Inicialmente `puestos_libres= S`. Si `puestos_libres=0` entonces el cliente se marcha, en caso contrario dicha variable debe ser decrementada en una unidad. Cuando el cliente termine de comer y se levante para irse la variable se incrementará en una unidad. Puesto que se trata de una variable que puede ser leída o escrita por diferentes procesos clientes es necesario asegurar la exclusión mutua en su uso. Para ello se va a utilizar el semáforo binario `ex_mut1` que se inicializa con el valor 1.

- Una vez que el cliente coge lo que quiere comer se pone en una cola para pagar, esta situación se puede modelar con un semáforo binario de nombre `a_pagar`. Inicialmente se inicializa a 1, ya que el primero que llega no tiene que esperar cola.

- Cada cliente debe avisar al dependiente para que le cobre, ya que este puede estar realizando tareas de reposición. Por tarea de reposición se entiende por ejemplo sacar un producto de una caja y colocarlo en un mostrador, llevar una caja de un sitio a otros, etc. Una tarea de reposición se ejecuta rápidamente, por lo que cuando el dependiente es avisado por el cliente, bien de viva voz o pulsando un llamador, el dependiente tarda poco en estar listo para cobrarle.

- Nótese que el dependiente no permanece sin hacer nada, es decir, bloqueado en espera de que le avise un cliente, sino que realiza una espera activa. Por ello no es posible utilizar un semáforo para avisar al cocinero. Se puede utilizar una variable global boleana de nombre `aviso_dependiente`, el cliente si desea ser atendido debe hacer

`aviso_dependiente=true`. El dependiente cuando termina de atender a un cliente hace `aviso_dependiente=false`. Inicialmente está variable se configura a `false`, ya que el dependiente está haciendo tareas de reposición. Puesto que se trata de una variable que puede ser leída o escrita por el clientes en el mostrador o el dependiente es necesario asegurar la exclusión mutua en su uso. Para ello se va a utilizar el semáforo binario `ex_mut2` que se inicializa con el valor 1.

- Finalmente es necesario usar un semáforo binario en el que espere el cliente hasta ser cobrado por el dependiente. Dicho semáforo se va a denotar como `finalizar_pago`, que debe inicializar a 0, el cliente debe hacer una operación `esperar(finalizar_pago)` mientras que corresponde al dependiente hacer una operación `señal(finalizar_pago)`.

```
program/module autoservicio;
var
      puestos_libres: integer;
      aviso_dependiente: boolean;
      a_pagar, ex_mut1, ex_mut2, finalizar_pago: semáforo;

process cliente;
begin
      /* Entra en el restaurante*/
      esperar(ex_mut1);
      if puestos_libres>0 then
            begin
                  /*Dejar pertenencias en el puesto para reservarlo*/
                  puestos_libres:= puestos_libres -1;
                  señal(ex_mut1);
                  /*Coger comida*/
                  esperar(a_pagar);
                  esperar(ex_mut2)
                  aviso_dependiente:= true;
                  señal(ex_mut2);
                  esperar(finalizar_pago);
                  señal(a_pagar);
                  /*Comer*/
                  esperar(ex_mut1);
                  puestos_libres:= puestos_libres +1;
                  señal(ex_mut1);
                  /*Abandona el restaurante*/
            end
      else
            begin
                  señal(ex_mut1);
                  /*Abandona el restaurante*/
            end
end
```

```
process dependiente;
begin
      loop
            esperar(ex_mut2);
            if aviso_dependiente==true then
                  begin
                        /*Cobrar cliente*/
                        señal(finalizar_pago);
                        aviso_dependiente:= false;
                        señal(ex_mut2);
                  end
            else
                  begin
                        señal(ex_mut2);
                        /* Hacer una tarea de reposición*/
                  end
            end
      end
end


begin
      puestos_libres :=S;
      aviso_dependiete :=false;
      inicializa (a_pagar,1);
      inicializa (ex_mut1,1);
      inicializa (ex_mut2,1);
      inicializa (finalizar_pago,0);
      cobegin
            clientes, dependiente;
      coend
end autoservicio;
```