Posibles soluciones a los ejercicios del parcial práctico de memoria compartida del 25-11-24

Importante: las soluciones que se muestran a continuación no son las únicas que se pueden considerar correctas para los ejercicios planteados.

1. Resolver con **SEMÁFOROS** el siguiente problema. La Clave Única de Identificación Tributaria (CUIT) es una clave que se utiliza en el sistema tributario de la República Argentina para poder identificar correctamente a las personas físicas o jurídicas. Consta de un total de once (11) cifras numéricas, siendo la última un dígito verificador (del 0 al 9). Una empresa cuenta con una lista de CUITs que debe procesar, debiendo informar la cantidad de CUITs por dígito verificador. Para ello, dispone de un software que emplea 5 workers, los cuales trabajan colaborativamente procesando de a una CUIT por vez cada uno. Al finalizar el procesamiento, el último worker en terminar debe informar los resultados del procesamiento. Notas: la función *obtenerDV(CUIT)* retorna el dígito verificador para la CUIT recibida como entrada. La lista de CUITs se almacena como una cola global y la solución debe maximizar la concurrencia.

```
sem sem_lista = 1;
sem sem_contDV [10] =  [10] {1};
sem sem_bar = 1;
Queue lista; // Ya cargada
int contadoresDV [10] = [10] {0};
int cant = 0;

Process worker[i=1..5] {
      int cuit, dv;
      // al ser una estructura compartida, la lista debe ser manipulada con exclusión
mutua
      P(sem_lista);
      while (not empty(lista)) do
            cuit = pop(lista);
            V(sem_lista);
            // obtener digito verificador
            dv = ObtenerDV(cuit);
            // con exclusión mutua, sumo al contador correspondiente sin afectar al resto
            P(sem_contDV[dv]);
            contadoresDV[dv]++;
            V(sem_contDV[dv]);
            P(sem_lista);
      end;
      V(sem_lista);
      // asegurarnos que todos terminaron antes de informar
      P(s_bar);
      cant = cant + 1;
      if (cant == 5) {
            for i := 0 to 9 do
                  print(i,contadoresDV[i]);
      }
      V(s_bar);
}
```

2. Resolver con **MONITORES** la siguiente situación. En un negocio hay UN empleado que diseña tarjetas digitales. El empleado debe atender los pedidos de C clientes, de acuerdo con el orden en que se hacen los pedidos. El cliente envía las indicaciones, y el empleado en base a eso diseña la tarjeta y se la envía al cliente. Notas: maximizar la concurrencia; existe una función *HacerTarjeta(indicaciones)* que simula el armado de la tarjeta por parte del empleado; todos los procesos deben terminar su ejecución.

```
Process Cliente[id: 0..C-1]
{ text instrucciones, tarjeta;
  instrucciones = GenerarInstrucciones();
  Negocio.Pedido (id, instrucciones, tarjeta);
}

Process Empleado
{ text inst, res;
  int idC;
  for (i=0; i<C; i++)
    { Negocio.Siguiente (idC, inst);
      res = HacerTarjeta(inst);
      Negocio.Resultado (idC, res);
    }
}

MONITOR Negocio
{  cola solicitudes;
   text resultados[C];
   cond empl, espera;

   Procedure Pedido (id: int, datos: text, res: OUT text)
   { push (solicitudes, (id, datos));
     signal (empl);
     wait (espera);
     res = resultados[id];
   }

   Procedure Siguiente (id: OUT int, datos: OUT text)
   { if ( empty(solicitudes) ) wait (empl);
     pop (solicitudes, (id, datos));
   }

   Procedure Resultado (id: int, res: text)
   { resultados[id] = res;
     signal (espera);
   }
}
```