Posibles soluciones a los ejercicios del parcial práctico del 10-11-25

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

TEMA 1

1. En un estadio de fútbol se dará un recital al que asistirán P personas. Para ello, cada asistente debe retirar su entrada por alguna de las 3 boleterías del estadio presentando su DNI (cada persona retira una única entrada). Los asistentes son atendidos por orden de llegada por alguno de los 3 empleados del club (cada empleado atiende una boletería). Implemente una solución usando **PMA**, considerando que los empleados atienden infinitamente. **Notas:** maximizar la concurrencia. La función *obtenerEntrada(dni)* le retorna al empleado la entrada para el DNI provisto por argumento; la función *obtenerDNI()* retorna el DNI para la persona que la invoca.

```
chan pedirEntrada(int,int);
chan recibirEntrada[P](Entrada);

Process Asistente [id=0..P-1] {
      int dni = obtenerDNI();
      Entrada ent;
      // enviar a canal único
      send pedirEntrada(id,dni);
      // recibir en canal específico
      receive recibirEntrada[id] (ent)
}

Process Empleado [id=0..2] {
      int dni, idAsis;
      Entrada ent;
      while (true) {
            receive pedirEntrada(idAsis,dni);
            ent = obtenerEntrada(dni);
            send recibirEntrada[idAsis] (ent);
      }
}
```

2. En un gimnasio hay una bicicleta de alta gama (que sólo puede ser usada por una persona a la vez) y un entrenador que controla su uso. También hay P deportistas que deben usar la bicicleta una única vez. Implemente una solución usando **PMS** considerando que el entrenador da acceso según el orden de llegada.

```
Process Deportista [id=0..P-1] {
      Buffer!pedirBicicleta(id);
      Entrenador?usarBicicleta();
      // usa la bicicleta
      Entrenador!liberarBicicleta();
}

Process Entrenador {
      int idDep;
      for i in 1..P {
            Buffer!pedirDeportista();
            Buffer?asignarDeportista(idDep);
            Deportista[idDep]!usarBicicleta();
            Deportista[idDep]?liberarBicicleta();
      }
}

Process Buffer {
      Queue deportistas;
      int idDep;
      for i in 1..2*P {
            if (Deportista[*] ? pedirBicicleta (idDep)) ->
                  push(deportistas,idDep);
            [] (not(empty(deportistas)); Entrenador ? pedirDeportista()) ->
                  idDep = pop(deportistas);
                  Entrenador!asignarDeportista (idDep);
      }
}
```

Comentario: otra solución adecuada consiste en combinar Entrenador y Buffer en un proceso único, considerando que su única tarea sería administrar el acceso a la bicicleta.

3. Implemente una solución para el siguiente problema usando **ADA**. En la farmacia de un hospital hay una única ventanilla para dispensar medicamentos. Los pacientes internados deben retirar sus medicamentos presentando una receta (un único retiro por persona). Los pacientes son atendidos en el orden de llegada, pero con prioridad de atención: los pacientes de urgencia tienen prioridad sobre los de internación general, y ambos tienen prioridad sobre los pacientes ambulatorios. Cada paciente entrega su receta y recibe la medicación correspondiente. **Notas:** la función *obtenerPrioridad()* le permite al paciente conocer su prioridad (retorna 0 si el paciente es de urgencia, 1 si es internado general, o 2 si es ambulatorio). La función *obtenerReceta()* retorna la receta para el paciente que la invoca mientras que *dispensarMedicamento(receta)* le retorna al empleado el medicamento para la receta indicada. El empleado atiende de a un paciente por vez, según la prioridad indicada. Todas las tareas deben finalizar.

```
Procedure Hospital IS

TASK Type Paciente;

TASK Empleado IS
      entry atencionUrgencia (rec: in Receta, med: out Medicamento);
      entry atencionInternacion (rec: in Receta, med: out Medicamento);
      entry atencionAmbulatorio (rec: in Receta, med: out Medicamento);
end Empleado;

pacientes: array (1..P) of Paciente;

TASK Body Paciente IS
      prioridad: integer;
      rec: Receta;
      med: Medicamento;
Begin
      rec = obtenerReceta()
      prioridad = obtenerPrioridad(),
      if (prioridad = 0) then
            Empleado.atencionUrgencia(rec,med);
      elsif (prioridad = 1) then
            Empleado.atencionInternacion(rec,med);
      else
            Empleado.atencionAmbulatorio(rec,med);
    end if;
end Paciente;

TASK Body Empleado IS
Begin
      for i in 1..P loop
            SELECT
                        Accept atencionUrgencia (rec: in Receta, med: out Medicamento)
do
                              -- atender paciente con urgencia
                              med = dispensarMedicamento(rec);
                        End atencionUrgencia;
                  OR    WHEN (atencionUrgencia'count = 0) =>
                        Accept atencionInternacion (rec: in Receta, med: out
Medicamento) do
                              -- atender paciente de internación
                              med = dispensarMedicamento(rec);
                        End atencionInternacion;
                  OR WHEN (atencionUrgencia'count = 0) AND (atencionInternacion'count =
0) =>
                        Accept atencionAmbulatorio (rec: in Receta, med: out
Medicamento) do
                              -- atender resto de pacientes
                              med = dispensarMedicamento(rec);
                        End atencionAmbulatorio;
            end SELECT;
      End loop;
End Empleado;
begin
      null;
end Hospital;
```

TEMA 2

1) En un estadio de fútbol se dará un recital al que asistirán P personas. Para ello, cada asistente debe retirar su entrada por alguna de las 3 boleterías del estadio presentando su DNI (cada persona retira una única entrada). Los asistentes son atendidos por orden de llegada por alguno de los 3 empleados del club (cada uno atiende una boletería). Implemente una solución usando **PMA**, considerando que los empleados se retiran cuando ya no hay personas por atender. **Notas:** maximizar la concurrencia. La función *obtenerEntrada(dni)* le retorna al empleado la entrada para el DNI provisto por argumento; la función *obtenerDNI()* retorna el DNI para la persona que la invoca. Todos los procesos deben terminar.

```
chan pedirEntrada(int,int);
chan recibirEntrada[P](Entrada);
chan pedirTrabajo(int);
chan asignarTrabajo[3](boolean,int,int);


Process Asistente [id=0..P-1] {
      int dni = obtenerDNI();
      Entrada ent;
      // enviar a canal único
      send pedirEntrada(id,dni);
      // recibir en canal específico
      receive recibirEntrada[id] (ent)
}

Process Admin {
      int idAsis, dni, idEmp;
      for i in 0..P-1 {
            receive pedirEntrada(idAsis,dni);
            receive pedirTrabajo(idEmp);
            send asignarTrabajo[idEmp] (true,idAsis,dni);
      }
      for i in 0..2 {
            receive pedirTrabajo(idEmp);
            send asignarTrabajo[idEmp] (false,NULL,NULL);
      }

}

Process Empleado [id=0..2] {
      int dni, idAsis;
      Entrada ent;
      boolean hayTrabajo;
      send pedirTrabajo(id);
      receive asignarTrabajo[id] (hayTrabajo,idAsis,dni)
      while (hayTrabajo) {
            ent = obtenerEntrada(dni);
            send recibirEntrada[idAsis] (ent);
            send pedirTrabajo(id);
            receive asignarTrabajo[id] (hayTrabajo,idAsis,dni);
      }
}
```

2) En un gimnasio hay una bicicleta de alta gama (que sólo puede ser usada por una persona a la vez) y un entrenador que controla su uso. También hay P deportistas que deben usar la bicicleta una única vez. Implemente una solución usando **PMS** considerando que el entrenador da acceso según el identificador del deportista (hasta que el deportista *i* no lo haya usado, el deportista *i+1* debe esperar).

```
Process Deportista [id=0..P-1] {
      Entrenador!pedirBicicleta();
      // usa la bicicleta
      Entrenador!liberarBicicleta();
}

Process Entrenador {
      int idDep;
      for i in 0..P-1 {
            Deportista[i]?pedirBicicleta();
            Deportista[i]?liberarBicicleta();
      }
}
```

3) Implemente una solución para el siguiente problema usando **ADA**. En un aeropuerto internacional hay una mesa de informes que atiende las consultas de los pasajeros. Existen P pasajeros que consultan una única vez, cada uno con una prioridad diferente: los pasajeros con vuelos inmediatos tienen prioridad sobre los que viajan más tarde; mientras que los pasajeros con movilidad reducida tienen prioridad sobre todos los demás. Cada pasajero entrega su código de vuelo al empleado de informes, quien responde con la puerta de embarque y la hora estimada de salida. **Notas:** Cada pasajero obtiene su prioridad mediante la función *obtenerPrioridad()*, que retorna 0 para movilidad reducida, 1 para vuelos inmediatos y 2 para el resto; mientras que la función *obtenerVuelo()* le retorna el código de vuelo. El empleado dispone de las funciones *obtenerPuerta(vuelo)* y *obtenerHora(vuelo)* que retornan la información de salida para el vuelo recibido. El empleado de informes atiende un pasajero por vez, según la prioridad indicada. Todas las tareas deben finalizar.

```
Procedure Aeropuerto IS

TASK Empleado IS
      entry atencionMoviReducida (vuelo: in string, puerta: out int, hora: out time);
      entry atencionInmediato (vuelo: in string, puerta: out int, hora: out time);
      entry atencionResto (vuelo: in string, puerta: out int, hora: out time);
end Empleado;

TASK Type Pasajero;
pasajeros: array (1..P) of Pasajero;

TASK Body Pasajero IS
      prioridad: integer;
      puerta: integer;
      vuelo: string;
      hora: time;
Begin
      vuelo = obtenerVuelo()
      prioridad = obtenerPrioridad(),
      if (prioridad = 0) then
            Empleado.atencionMoviReducida(vuelo,puerta,hora);
      elsif (prioridad = 1) then
            Empleado.atencionInmediato(vuelo,puerta,hora);
      else
            Empleado.atencionResto(vuelo,puerta,hora);
    end if;
end Pasajero;

TASK Body Empleado IS
Begin
      for i in 1..P loop
            SELECT
                        Accept atencionMoviReducida (vuelo: in string, puerta: out int,
hora: out time) do
                              -- atender pasajero con movilidad reducida
                              puerta = ObtenerPuerta(vuelo); hora = ObtenerHora(vuelo);
                        End atencionMoviReducida;
                  OR    WHEN (atencionMoviReducida'count = 0) =>
                        Accept atencionInmediato (vuelo: in string, puerta: out int,
hora: out time) do
                              -- atender pasajero de vuelo inmediato
                              puerta = ObtenerPuerta(vuelo); hora = ObtenerHora(vuelo);
                        End atencionInmediato;
                  OR WHEN (atencionMoviReducida'count = 0) AND (atencionInmediato'count
= 0) =>
                        Accept atencionResto (vuelo: in string, puerta: out int, hora:
out time) do
                              -- atender resto de pasajeros
                              puerta = ObtenerPuerta(vuelo); hora = ObtenerHora(vuelo);
                        End atencionResto;
            end SELECT;
      End loop;
End Empleado;
begin
      null;
end Aeropuerto;
```