Práctica 5 – Rendezvous (ADA)

CONSIDERACIONES PARA RESOLVER LOS EJERCICIOS:

1. NO SE PUEDE USAR VARIABLES COMPARTIDAS
2. Declaración de tareas
   - Especificación de tareas sin ENTRY's (nadie le puede hacer llamados).
     TASK Nombre;
     TASK TYPE Nombre;
   - Especificación de tareas con ENTRY's (le puede hacer llamados). Los entry's funcionan de manera semejante a los procedimientos: solo pueden recibir o enviar información por medio de los parámetros del entry. NO RETORNAN VALORES COMO LAS FUNCIONES
     TASK [TYPE] Nombre IS
     ENTRY e1;
     ENTRY e2 (p1: IN integer; p2: OUT char; p3: IN OUT float);
     END Nombre;
   - Cuerpo de las tareas.
     TASK BODY Nombre IS
     Codigo que realiza la Tarea;
     END Nombre;
3. Sincronización y comunicación entre tareas
   - ***Entry call*** para enviar información (o avisar algún evento).
     NombreTarea.NombreEntry (parametros);
   - ***Accept*** para atender un pedido de entry call sin cuerpo (sólo para recibir el aviso de un evento para sincronización).
     ACCEPT NombreEntry (p1: IN integer; p3: IN OUT float);
   - ***Accept*** para atender un pedido de entry call con cuerpo.
     ACCEPT NombreEntry (p1: IN integer; p3: IN OUT float) do
     Cuerpo del accept donde se puede acceder a los parámetros p1 y p3.
     Fuera del entry estos parámetros no se pueden usar.
     END NombreEntry;
   - El accept se puede hacer en el cuerpo de la tarea que ha declarado el entry en su especificación. Los entry call se pueden hacer en cualquier tarea o en el programa principal.
   - Tanto el entry call como el accept son bloqueantes, ambas tareas continúan trabajando cuando el cuerpo del accept ha terminado su ejecución.

4. Select para ENTRY CALL.
   - Select ...OR DELAY: espera a lo sumo x tiempo a que la tarea correspondiente haga el accept del entry call realizado. Si pasó el tiempo entonces realiza el código opcional.

```
SELECT
    NombreTarea.NombreEntry(Parametros);
    Sentencias;
OR DELAY x
    Código opcional;
END SELECT;
```

   - Select ...ELSE: si la tarea correspondiente no puede realizar el accept inmediatamente (en el momento que el procesador está ejecutando esa línea de código) entonces se ejecuta el código opcional.

```
SELECT
    NombreTarea.NombreEntry(Parametros);
    Sentencias;
ELSE
    Código opcional;
END SELECT;
```

   - En los select para entry call sólo puede ponerse un entry call y una única opción (OR DELAY o ELSE);

5. Select para ACCEPT.
   - En los select para los accept puede haber más de una alternatica de accept, pero no se puede haber alternativas de entry call (no se puede mezclar accept con entries). Cada alternativas de ACCEPT puede ser o no condicional (WHEN).

```
SELECT
    ACCEPT e1 (parámetros);
    Sentencias1;
OR
    ACCEPT e2 (parámetros) IS cuerpo; END e2;
OR
   WHEN (condición) => ACCEPT e3 (parámetros) IS cuerpo; END e3;
   Sentencias3
END SELECT;
```

     Funcionamiento: Se evalúa la condición booleana del WHEN de cada alternativa (si no lo tiene se considera TRUE). Si todas son FALSAS se sale del select. Sino, de las alternativas cuyo condición es verdadera se elige en forma no determinística una que pueda ejecutarse inmediatamente (es decir que tiene un entry call pendiente). Si ninguna de ellas se puede ejecutar inmediatamente el select se bloquea hasta que haya un entry call para alguna alternativa cuya condición sea TRUE.

   - Se puede poner una opción OR DELAY o ELSE.
   - Dentro de la condición booleana de una alternativa (en el WHEN) se puede preguntar por la cantidad de entry call pendientes de cualquier entry de la tarea.

```
NombreEntry'count
```

   - Después de escribir una condición por medio de un WHEN siempre se debe escribir un accept.

1. Se requiere modelar un puente de un solo sentido, el puente solo soporta el peso de 3 autos, 2 camionetas o un camión.

2. Se quiere modelar la cola de un banco que atiende un solo empleado, los clientes llegan y si esperan más de 10 minutos se retiran.

3. Se debe modelar un buscador para contar la cantidad de veces que aparece un número dentro de un vector distribuido entre las ***N*** tareas ***contador***. Además existe un administrador que decide el número que se desea buscar y se lo envía a los ***N contadores*** para que lo busquen en la parte del vector que poseen.

4. Se debe controlar el acceso a una base de datos. Existen A procesos de Tipo 1, B procesos de Tipo 2 y C procesos de Tipo 3 que trabajan indefinidamente de la siguiente manera:
   - Proceso Tipo 1: intenta escribir, si no lo logro en 2 minutos, espera 5 minutos y vuelve a intentarlo.
   - Proceso Tipo 2: intenta escribir, si no lo logra en 5 minutos, intenta leer, si no lo logra en 5 minutos vuelve a comenzar.
   - Proceso Tipo 3: intenta leer, si no puede inmediatamente entonces espera hasta poder escribir.

   Un proceso que quiera escribir podrá acceder si no hay ningún otro proceso en la base de datos, al acceder escribe y avisa que termino de escribir. Un proceso que quiera leer podrá acceder si no hay procesos que escriban, al acceder lee y avisa que termino de leer. Siempre se le debe dar prioridad al pedido de acceso para escribir sobre el pedido de acceso para leer.

5. Se dispone de un sistema compuesto por ***1 central*** y ***2 procesos***. Los procesos envían señales a la central. La central comienza su ejecución tomando una señal del proceso 1, luego toma aleatoriamente señales de cualquiera de los dos indefinidamente. Al recibir una señal de proceso 2, recibe señales del mismo proceso durante 3 minutos.
   El proceso 1 envía una señal que es considerada vieja (se deshecha) si en 2 minutos no fue recibida.
   El proceso 2 envía una señal, si no es recibida en ese instante espera 1 minuto y vuelve a mandarla (no se deshecha).

6. Suponga que existen ***N usuarios*** que deben ejecutar su programa, para esto comparten ***K procesadores***.
   Los ***usuarios*** solicitan un procesador al administrador. Una vez que el administrador les entregó el número de procesador, el usuario le da su programa al procesador que le fue asignado. Luego el usuario espera a que:
   - El procesador le avise si hubo algún error en una línea de código con lo cual el usuario arregla el programa y se lo vuelve a entregar al procesador, es decir queda nuevamente en la cola de programas a ejecutar por su procesador. El

usuario no termina hasta que el procesador haya ejecutado su programa correctamente (sin errores).
- El procesador le avise que su programa termino, con lo cual termina su ejecución.

El ***administrador*** tomará los pedidos de procesador hechos por los usuarios y balanceara la carga de programas que tiene cada procesador, de esta forma le entregará al usuario un número de procesador.
El ***procesador*** ejecutará un Round-Robin de los programas listos a ejecutar. Cada programa es ejecutado línea por línea por medio de la función *EJECUCIÓN* la cual devuelve:
- 1 error en la ejecución.
- 2 normal.
- 3 fin de programa.

NOTA: Suponga que existe también la función *LineaSiguiente* que dado un programa devuelve la línea a ser ejecutada. Maximice la concurrencia en la solución.

7. En una clínica existe un ***médico*** de guardia que recibe continuamente peticiones de atención de las ***E enfermeras*** que trabajan en su piso y de las ***P personas*** que llegan a la clínica ser atendidos.
   Cuando una *persona* necesita que la atiendan espera a lo sumo 5 minutos a que el médico lo haga, si pasado ese tiempo no lo hace, espera 10 minutos y vuelve a requerir la atención del médico. Si no es atendida tres veces, se enoja y se retira de la clínica.
   Cuando una *enfermera* requiere la atención del médico, si este no lo atiende inmediatamente le hace una nota y se la deja en el consultorio para que esta resuelva su pedido en el momento que pueda (el pedido puede ser que el médico le firme algún papel). Cuando la petición ha sido recibida por el médico o la nota ha sido dejada en el escritorio, continúa trabajando y haciendo más peticiones.
   El *médico* atiende los pedidos teniendo dándoles prioridad a los enfermos que llegan para ser atendidos. Cuando atiende un pedido, recibe la solicitud y la procesa durante un cierto tiempo. Cuando está libre aprovecha a procesar las notas dejadas por las enfermeras.
   NOTA: maximice la concurrencia.