Parcial Práctico de MC - Primera Fecha - 11/10/2022

1. Resolver con ***SEMÁFOROS*** el siguiente problema. En una planta verificadora de vehículos, existen ***7 estaciones*** donde se dirigen 150 vehículos para ser verificados. Cuando un vehículo llega a la planta, el ***coordinador*** de la planta le indica a qué estación debe dirigirse. El coordinador selecciona la estación que tenga menos vehículos asignados en ese momento. Una vez que el vehículo sabe qué estación le fue asignada, se dirige a la misma y espera a que lo llamen para verificar. Luego de la revisión, la estación le entrega un comprobante que indica si pasó la revisión o no. Más allá del resultado, el vehículo se retira de la planta. ***Nota:*** maximizar la concurrencia.

2. Resolver con **MONITORES** el siguiente problema. En un sistema operativo se ejecutan ***20 procesos*** que periódicamente realizan cierto cómputo mediante la función *Procesar()*. Los resultados de dicha función son persistidos en un archivo, para lo que se requiere de acceso al subsistema de E/S. Sólo un proceso a la vez puede hacer uso del subsistema de E/S, y el acceso al mismo se define por la prioridad del proceso (menor valor indica mayor prioridad).

[redacted]

HOJA N° 1

FECHA 11/10/22

(1) Sem esperaV[150] = ([150] 0), mutex = 1; esperaCoord = 0; mutexPlanta = 1; plantas[7] = ([7] 0);
cola fila; plantas_cant[7] = ([7] 0); asignados[150] = ([150] 0); Cola[7] fila_plantas; //Array de Colas int
boolean FinDia = false; text[150] resultados;

```
Process Vehiculo [id: 0..149] { idPlanta int;
  revision: text;
  P(mutex)
  fila.push(id); V(mutex);
  V(esperaCoord); //avisa a coordinador
  P(esperaV[id]); //espera asignen planta
  //Dirige a planta
  idPlanta = asignados[id];
  P(mutexPlanta)
  fila_plantas[idPlanta].push(id);
  V(mutexPlanta);
  V(plantas[idPlanta]);
  P(esperaV[id]);
  revision = resultados[id];
  //salir
}
```

```
Process Coordinador {
  idV int;
  for(int i = 0..149) {
    P(esperaCoord); // llego un vehiculo
    P(mutex);
    idV = fila.pop();
    V(mutex)
    P(mutexPlanta);
    idPlanta = obtenerPrimeraConMenos(plantas_cant);
    plantas_cant[idPlanta]++;
    V(mutexPlanta);
    asignados[idV] = idPlanta;
    V(esperaV[id]);
  }
  P(mutex)
  finDia = true;
  V(mutex);
  for (int i=0..6) { V(planta[id]); }
}
```

```
Process Planta [id: 0..6] {
  P(mutex)
  while (not finDia) {
    V(mutex),
    P(plantas[idPlanta]);
    P(mutexPlanta)
    if (fila_planta[id].empty() == false) {
      v = fila_plantas[id].pop();
      V(MutexPlanta);
      resultados[v] = verificar(); //devuelve el resultado
      V(esperaV[v]);
      P(mutexPlanta);
      planta_cant[id]--;
      V(mutexPlanta);
    }
    P(mutex);
  }
  V(mutex);
}
```

②

```
Procedure Proceso [id: 0..19] {
  int prioridad = random(0,10);
  r = Procesar();
  Sistema.llegue(prioridad, id);
  guardar En Archivo (r);
  Sistema.salir();
}
```

```
/* Tupla (e1, e2)..
   Acceso e1 = tup
          e2 = tup

Monitor Sistema {
  cond esperar[10];
  cola esperando
  int sem = 1

  Procedure llegue (prioridad: int, id: int)

    if (sem == 0) {
        esperando.InsertarEnOrden((id, priorid
        wait (esperar[id]);
    } else {
      sem--;
    }
  }


  Procedure salir () {
    if (esperando.empty() == false) {
      tupla = esperando.pop()
      idAux = tupla[0] //Accedo al id
      signal (esperar[idAux]);
    } else {
      sem ++
    }
  }
```