Práctica de repaso 2

1. Resolver los problemas siguientes:
   a) En una estación de trenes, asisten P personas que deben realizar una carga de su tarjeta SUBE en la terminal disponible. La terminal es utilizada en forma exclusiva por cada persona de acuerdo con el orden de llegada. Implemente una solución utilizando únicamente procesos Persona. Nota:la función UsarTerminal() le permite cargar la SUBE en la terminal disponible.

```
sem mutex=1;
sem sig[P]= ([P] 0);
boolean libre=true;
cola fila;

process Persona [id: 0.. P-1]{
    P(mutex);
    if (!libre){
        fila.push(id);
        V(mutex);
        P(sig[id]);
    }
    else{
        libre=false;
        V(mutex);
    }
    usarTerminal();
    P(mutex);
    if (!fila.vacia()){
        V(sig[fila.pop]);
    else
        libre= true;
    V(mutex);
    }
```

b) Resuelva el mismo problema anterior pero ahora considerando que hay T terminales disponibles. Las personas realizan una única fila y la carga la realizan en la primera terminal que se libera. Recuerde que sólo debe emplear procesos Persona. Nota: la función UsarTerminal(t) le permite cargar la SUBE en la terminal t.

```
sem mutex=1
sem sig[P]= ([P] 0);
cola fila;
cola terminal;
vector terminal[P];

process Persona [id: 0.. P-1]{
    int siguiente;
    int t;
    P(mutex);
    if (terminal.vacia()){
        fila.push(id);
        V(mutex);
        P(sig[id]);
        usarTerminal(terminal[id]);
        }
    else{
      t= terminal.pop();
        V(mutex)
        usarTerminal(t);
    }
        P(mutex);
        if (!fila.vacia()){
            siguiente=fila.pop();
            terminal[siguiente]= t ;
            V(sig[siguiente]);
        }
        else
```

```
            terminal.push(t);
        V(mutex);
    }
```

2. Implemente una solución para el siguiente problema. Un sistema debe validar un conjunto de 10000 transacciones que se encuentran disponibles en una estructura de datos. Para ello, el sistema dispone de 7 workers, los cuales trabajan colaborativamente validando de a 1 transacción por vez cada uno.
   Cada validación puede tomar un tiempo diferente y para realizarla los workers disponen de la
   función Validar(t), la cual retorna como resultado un número entero entre 0 al 9. Al finalizar el
   procesamiento, el último worker en terminar debe informar la cantidad de transacciones por cada resultado de la función de validación. Nota: maximizar la concurrencia.

```
sem mutex[10]=([10] 1);
vector transacciones [10000] contador[10];
sem fin=1;
int finalizados=0;

process Worker[id: 0.. 6]{
    vector contadorLocal[10];
    int valor;
    int parte= 10000/7;
    for i:= parte * id to parte { //suponiendo que es su part
        valor = Validar(transacciones [i]);
        contadorLocal[valor]+=1;
    }
    for i:= 1 to 10{
        P(mutex[i-1])
        contador[i-1]+= contadorLocal[i-1];
        V(mutex[i-1]);
```

```
    }
    P(fin);
    finalizados+=1;
    if (finalizados == 7)
        for i:= 1 to 10
            writeln(contador[i-1]);
    V(fin);
}
```

3. Implemente una solución para el siguiente problema. Se debe simular el uso de una máquina
   expendedora de gaseosas con capacidad para 100 latas por parte de U usuarios. Además, existe un repositor encargado de reponer las latas de la máquina. Los usuarios usan la máquina según el orden de llegada. Cuando les toca usarla, sacan una lata y luego se retiran. En el caso de que la máquina se quede sin latas, entonces le debe avisar al repositor para que cargue nuevamente la máquina en forma completa. Luego de la recarga, saca una botella y se retira. Nota: maximizar la concurrencia; mientras se reponen las latas se debe permitir que otros usuarios puedan agregarse a la fila.

```
cola fila;
sem sig[U]=([U] 0);
cola gaseosas;
boolean libre=true;
sem mutexC=1;

process Usuario[id: 0.. P-1]{
    elem gaseosa;
    P(mutexC);
    if(!libre){
        fila.push(id);
        V(mutexC);
        P(sig[id]);
```

```
    }
    else{
        libre:= false;
        V(mutexC);
    }
    if (gaseosas.vacia()){
        V(rellenar);
        P(esperar);
    }
    gaseosa= gaseosas.pop();
    P(mutexC)
    if (!fila.vacia())
        V(sig[fila.pop()]);
    else
        libre:= true;
    V(mutexC);
}


process Repositor(){
    while(true){
        P(rellenar);
        for i:= 1 to 100
            gaseosas.push(gaseosa);
        V(esperar);
    }
}
```