Practica 1

sábado, 19 de septiembre de 2020 12:13

1) a)
2)

```
int M = 10;
int N = 5;

int numeros[M];
int numero = N;
int total = 0;

process buscador [i = 0 to M]{
    if(numeros[i] == N){
        <total++;>
    }
}
```

3)
a)

```
Process Productor::
{ while (true)
    { produce elemento
    <await (cant < N)>
    buffer[pri_vacia] = elemento;
    <cant++>
    pri_vacia = (pri_vacia + 1) mod N;
    }
}

Process Consumidor::
{ while (true)
    { <await (cant > 0)>
    elemento = buffer[pri_ocupada];
    <cant-->
    pri_vacia = (pri_vacia + 1) mod N;
    consume elemento
    }
}
```

b)

```
int P = 5;
int C = 4;
int cant = 0;
int pri_ocupada = 0;
```

```
int P = 5;
int C = 4;
int cant = 0;
int pri_ocupada = 0;
int pri_vacia = 0;
int buffer[N];

Process Productor[i = 0 to P]
{ while (true)
    { produce elemento
    <await (cant < N);
    buffer[pri_vacia] = elemento;
    <cant++>
    pri_vacia = (pri_vacia + 1) mod N;>
    }
}

Process Consumidor[i = 0 to C]
{ while (true)
    { <await (cant > 0);
    elemento = buffer[pri_ocupada];
    <cant-->
    pri_vacia = (pri_vacia + 1) mod N;>
    consume elemento
    }
}
```

c)

```
c)
n = 10;
int personas = n;
boolean enUso = false;
personas cola;
boolean ready[n] = false;
int siguiente = n + 1;

process personas[i = 1 to n]{
    ready[i] = true;
    <await i == siguiente;>
    imprimir(documento);
    ready[i] = false;
}

process coordinador{
    while(true){
```

```
process coordinador{
    while(true){
        int j = 1;
        while(!ready[j-1] && j<=n){
            j++;
        }
        if(j < n){
            siguiente = j;
            <await ready[siguiente] = false>
        }
    }
}
```

4)

```
int pila[5];

Process usador[i = 0 to 5]
{ while (true)
    { <await (pila.length > 0);
    elem = pila.pop(pila);>
    usa el elemento;
    <pila.push(elem);>
    }
}
```

5)

```
a)
n = 10;
int personas = n;
boolean enUso = false;

process personas[i = 1 to n] {
    <await !enUso; enUso = true>
    Imprimir(documento);
    unUso = false;
}

b)
n = 10;
int personas = n;
boolean enUso = false;
personas cola;
int siguiente = 0;
int proximo = 1;
```

```
personas cola;
int siguiente = 0;
int proximo = 1;

process personas[i = 1 to n] {
    <int id = siguiente + 1; siguiente++;>
    <await id == proximo>
    imprimir(documento);
    proximo++;
}

c)
n = 10;
int personas = n;
boolean enUso = false;
personas cola;
boolean ready[n] = false;
int siguiente = n + 1;

process personas[i = 1 to n]{
    ready[i] = true;
    <await i == siguiente;>
    imprimir(documento);
    ready[i] = false;
}

process coordinador{
    while(true){
        int j = 1;
        while(!ready[j-1] && j<=n){
            j++;
        }
        if(j < n){
            siguiente = j;
            <await ready[siguiente] = false>
        }
    }
}

d)
n = 10;
int personas = n;
boolean enUso = false;
```

```
n = 10;
int personas = n;
boolean enUso = false;
personas cola;
int siguiente = 0;
int proximo = 0;

process personas[i = 1 to n] {
    <int id = siguiente + 1; siguiente++;>
    <await id == proximo;>
    imprimir(documento);
    enUso = false;
}

process coordinador{
    while(proximo < n){
        enUso = true;
        proximo++;
        <await !enUso>
    }
}
```

6) La condición que no cumple es la ausencia de demora innecesaria, ya que si el primer proceso tiene que entrar 2 veces seguidas no va a poder porque tiene que esperar al proceso dos.

7)

```
int arribo[1:n] = ([n]0);
int continuar[1:n]= ([n]0);

processWorker[i = 1 to n]{
    while(true){
        arribo[i]= 1;
        while (continuar[i] == 0) skip;
        //hace mambos críticos
        continuar[i] = 0;
    }
}

processCoordinador {
    while(true){
        for[i = 1 to n st arribo[i] == 1]{
```

```
    while(true){
        for[i = 1 to n st arribo[i] == 1]{
            arribo[i]= 0;
            continuar[i] = 1;
            while(continuar[i] == 0) skip;
        }
    }
}
```